51CTO.com
技术博客Blog

博客月刊

blog.51cto.com

2014年03月
总第 03 期

- 几个Hadoop部署必备的小脚本
- 几个负载均衡器的实例小结
- DG 主库fial over，强制激活备库解决案例
- 记一次奇葩的sleep引起的Too many connections
- 一个优秀IT系统管理员该有的良好习惯

目录

AD DS 最佳实践分析程序（BPA）应用实例

作者：虚拟化应用专家 方建国 来源：http://rickyfang.blog.51cto.com/1213/1364477

最近，在项目上做 Windows Server 2008 R2 AD DS 的健康检查，使用到了一个装完 AD DS 后系统自带的工具 AD DS 的最佳实践分析程序，发在博客上和各位分享下。

在 Windows Server 2008 R2 以后的版本中也有，本文是在 Windows Server 2008 R2 的 AD 域环境中使用的。

用法，估计大家都会用了，俺分享的主要目的是如何导出或者说是归档分析扫描的结果。

一、 AD DS BPA 适用的 Windows Server 2008 R2 的版本

AD DS BPA 可以在以下版本的 Windows Server 2008 R2 中使用，且为域控制器角色或 RODS 角色：

· Windows Server 2008 R2 Standard

· Windows Server 2008 R2 Enterprise

· Windows Server 2008 R2 Datacenter

二、 AD DS BPA 的规则和验证的条件

以下来源于微软 Technet 网站。

在 Windows Server 2008 R2 中，AD DS BPA 扫描验证下列 AD DS 配置设置：

· 域名系统 (DNS) 相关规则，这些规则验证所有条件中的下列条件：

o 域控制器能够访问与之相关的 DNS 服务器并检索 DNS 记录。 （这是前提规则。）

o 域控制器的所有必要域和林主机（A 或 AAAA）资源记录均已在 DNS 中注册。

o 域控制器的所有必要 DNS 主机（A 或 AAAA）资源记录均已在 DNS 中注册了正确的 IP 地址。

o 域控制器的所有必要的站点特定和全局服务（SRV）资源记录均已在 DNS 中注册。

o 域控制器的必要别名（CNAME）资源记录已在 DNS 中注册。

· 操作主机（也称为灵活单主机操作或 FSMO）连接规则（前提规则），该规则验证下列条件：

o 域控制器可以连接到该域中的相对 ID（RID）操作主机、基础结构操作主机和主域控制器（PDC）仿真器操作主机。

o 域控制器可以连接到该林中的架构操作主机和域命名操作主机。

· 操作主机角色所有权规则，该规则验证下列条件：

o 架构主机角色和域命名主机角色由林中的同一域控制器拥有。

o RID 主机角色和 PDC 仿真器主机角色由域中的同一域控制器拥有。

· 域中的控制器数目规则，用于验证下列条件：域至少拥有两个正在运行的域控制器。

三、 实例操作

实例操作的大致过程如下：

1、先运行图形界面的 AD DS BPA，以扫描出结果；

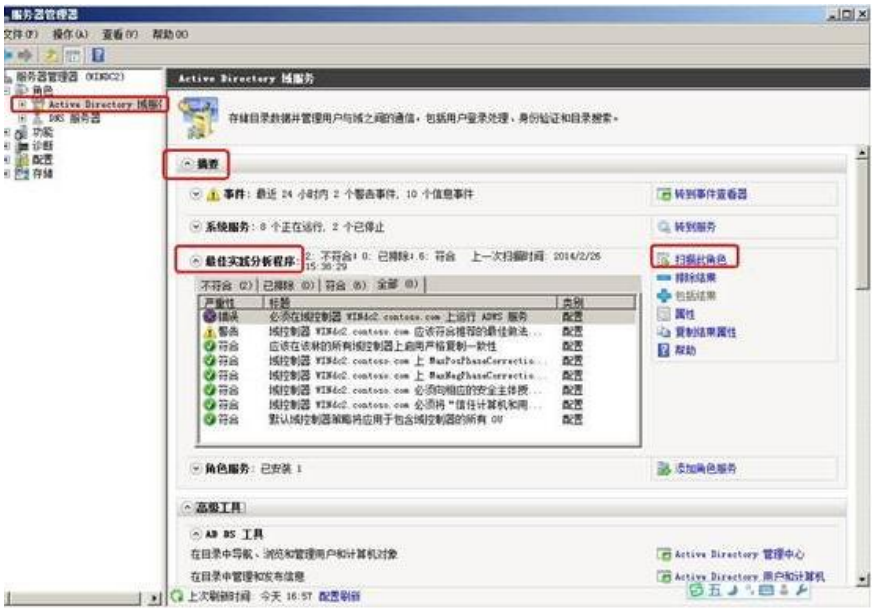
2、以管理员身份运行 PS，并加载相应的 BPA 模块；

3、列举出当前 AD 域上可用的 BPA 模块 ID；

4、从列举出的 BPA 模块中 ID 选择 AD DS 的模块 ID；

5、获取扫描结果并转换成 HTML 格式。

1、在域控制器上（装有 AD DS），打开服务器管理器，并导入至角色—Active Directory 域服务---摘要---最佳实践分析程序，并运行“扫描此角色”。



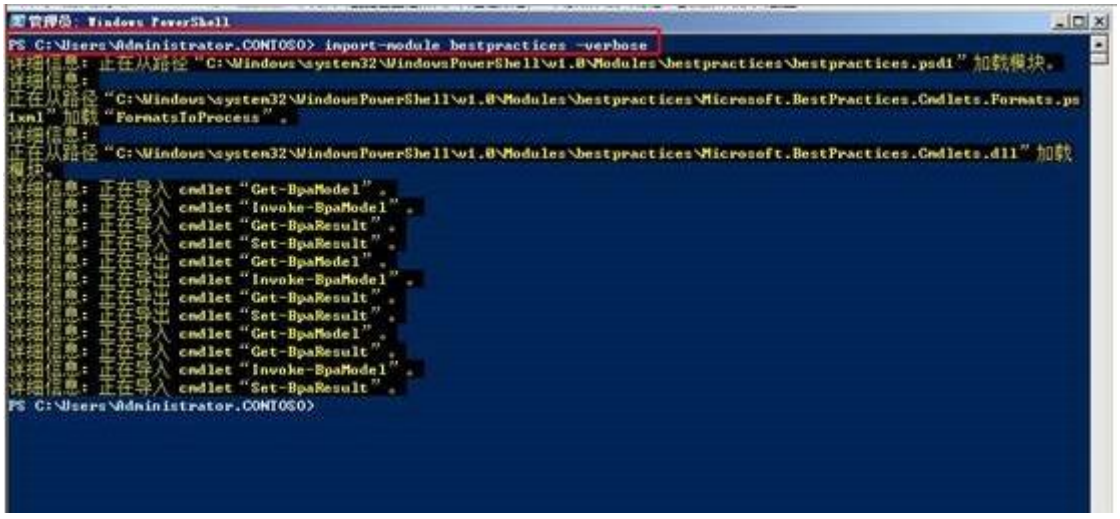
2、运行完后，如何取出结果呢，尤其是想导出 HTML 的文件，以备报告使用？这也是本文行成的真正目的所在。

（AD DS PS 的使用，分为两种，一是 UI 的方式，一是 PS 的方式。本文采用 UI 的方式运行，PS 方式归档并导出结果）

以管理员身份，打开 PS，运行以下命令，以加载相关模块：

Import-Module BestPractices -Verbose

(最佳操作实践分析工具是以 PowerShell 模块形式实现的，因此我们在使用时首先需要加载模块)



3、获取相关模块的 ID，这个是较重要的一步，一般不知道 BPA 有哪些模块，以及模板的名称是什么，就可以使用此命令来得到：

Get-BpaModel

（获得当前 PowerShell 中所有可用的最佳操作实践分析工具）

我们主要是用到"Microsoft/Windows/DirectoryServices"，也就是 AD DS BPA 的模块 ID。

4、 得到 AD DS BPA 的扫描结果，运行转换，使 CSS 文件转换成 HTML 文件，并保存到相应的位置：

Get-BPAResult –BestPracticesModelId "Microsoft/Windows/DirectoryServices" | ConvertTo-Html –As List –CssUri $env:windir\system32\WindowsPowerShell\v1.0\Modules\BestPractices\BestPracticesReportFormat.css > c:\dsbpa.html

5、使用 IE 浏览器，打开上一步中导出的 HTML 结果文件，就是如下图所示的内容了。

至此，也本文完成，再次提醒重点关注第 3 步、第 4 步。

利用 Ossim 系统进行主机漏洞扫描

作者：IT 系统架构专家 李晨光 来源：http://chenguang.blog.51cto.com/350944/1349749

企业中查找漏洞要付出很大的努力，不能简单的在服务器上安装一个漏洞扫描软件那么简单，那样起不了多大作用。这并不是因为企业中拥有大量服务器和主机设备，这些服务器和设备又通不同速率的网络互联，只是我们在期望的时间内无法获得所需的覆盖范围，目前许多欧美的国际安全组织都按照自己分类准则建立了各自的数据库其中主流是 CVE 和，XForce。

他的好处是，当网络出现安全事故，入侵检测系统(IDS)产生警报时，像 CVE 这类标准的系统脆弱性数据库网络安全工作就显得极为重要!，目前在中国国家计算机网络应急处理协调中心（CNCERT/CC）领导下，国内也组建了自己的 CVE 组织——CNCVE，CNCVE 的组建目的就是建设一个具有中国特色的，能为国内广大用户服务的 CVE 组织。但是并不是说拥有了 CVE 就囊括了所有漏洞问题，除了这些开放的脆弱性数据库外，还应该存在大量的没有对公众开放的脆弱性数据库。有的可能大家根本就不知道这些脆弱性数据库的存在。

1.CVE

CVE （Common Vulnerabilities and Exposures）是由美国国土安全部门（US Department Of Homeland Security，简称 DHS）成立，由非盈利组织 MITRE 公司管理和维护至今。

Vulnerability（漏洞，脆弱性）这个词汇可以有狭义和广义多种解释。比如说：finger 服务，可能为入侵者提供很多有用的资料，但是该服务本身有时是业务必须的，而且不能说该服务本身有安全问题。

CVE 标准命名

为方便独立的脆弱性数据库和不同安全工具彼此之间能够更好地共享数据 如下图脆弱性数据库所示。

CVE 的标准命名方式是由“CVE”、时间和编号共同组成。例如，命名为“CVE-2008-6021”的条目表示 2008 年第 6021 号脆弱性。

CVE 的内容是 CVE 编辑委员会的合作努力的成果。这个委员会的成员来自于许多安全相关的组织，如软件开发商、大学研究机构、政府组织和一些优秀的安全专家等，而且 CVE 可以免费阅读和下载。

图 CVE 脆弱性数据库

2.OSVDB

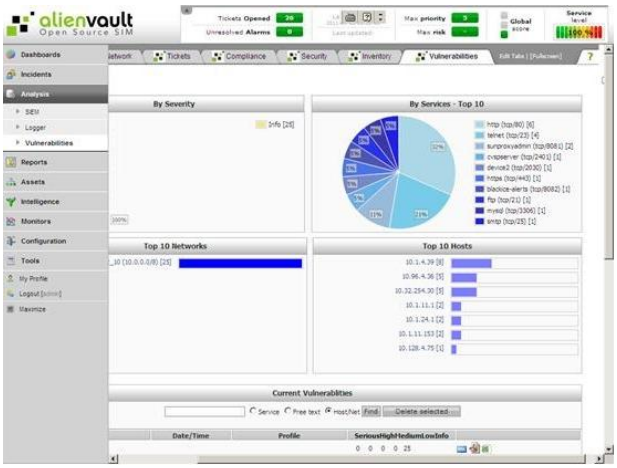
OSVDB （Open Source Vulnerability Database）是由一个社团组织创立并维护的独立开源的数据库。它最早是在 2002 年的 Black Hat 和 Defcon 安全会议上提出的一项服务，它提供了一个独立于开发商的脆弱性数据库实现方案。和 CVE 一样 OSVDB 数据库也是开源并且免费的。它由安全事业爱好者来维护，向个人和商业团体免费开放。两者的差异在于 CVE 提供标准名称，可以通俗理解为数据字典，而

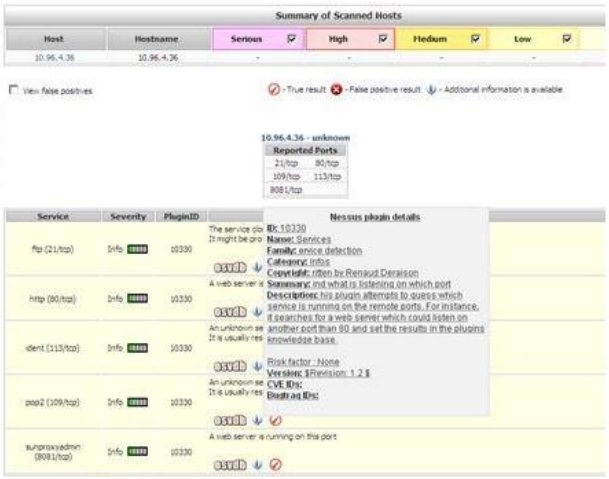
OSVDB 为每一条脆弱性提供了详尽的信息，OSVDB 需要参考 CVE 的名称。

3.BugTraq

BugTraq [3]是由 Security Focus 管理的 Internet 邮件列表，现在已被赛门铁克公司收购。在电脑安全世界，BugTraq 相当于最权威的专业杂志。大多数安全技术人员订阅 Bugtraq，因为这里可以抢先获得关于软件、系统漏洞和缺陷的信息，还可以学到修补漏洞和防御反击的招数。

接下来我们看看通过 ossim 系统中的 Openvas 来扫描系统漏洞的例子，工作界面截图所下图所示：





生产环境中高可用 DNS 方案

作者：倪增光 来源：http://caiguangguang.blog.51cto.com/1652935/1357867

昨天老毕跟我讨论 DNS 的问题，今天想了想，其实实际在生产环境中用到的 DNS 大体有 3 种：

1.内网调用 DNS

一般是每个机房一组，做 ms 结构，主要功能是供机房内部应用之间调用解析，减少对 hosts 的依赖，避免硬编码。一般 ms 结构就可以了，不太会做其他的高可用。

2.回源调用 DNS

和内网调用 DNS 功能差不多，不过这个是跨机房的调用，而且一般会通过 view 做智能 DNS 解析。比较常见的应用场景就是 webcdn 的回源 DNS，结构上和内部调用 DNS 差不多。

3.外网 DNS

提供各种针对外网的解析，A 记录，CNAME，MX 等等。

作为用户访问的入口，其重要性也不言而喻，因此对可用性和扩展性以及性能要求会比较高。 一般的做法是在 BGP 机房做一个高可用的集群做主要的解析工作，同时在双线或者单线机房做一组冷备的高可用 DNS 集群。

这里讨论下用 lvs 做 DNS 高可用的方案，主要借鉴了 mysql 高可用的方案（一般都是一个主库，后面挂多个从库，从库前面使用 lvs+keepalived 做 ha，程序写入主库，读取从库）。

具体的流程如下：

1.用户通过 web 前端来做 DNS 的增删改等操作（如果是 python 程序的话，可以通过 DNSpython 操作）。

2.slave 通过对比 zone 文件的 serial 值来决定是否需要更新记录。

3.用户查询请求至由 lvs(dr 模式)+keepalived 组成的前端负载均衡。

4.lvs 分发请求至 slave server,最后由 slave server 直接向用户响应请求。

由于 DNS 一般情况是用 udp 协议的（关于 DNS 使用的协议可以参考之前的 blog），而大家一般都是用 lvs 做 tcp 的 load balance.

在用 lvs 实现 udp 的 load balance 时主要注意以下几点：

1.添加 realserver 时选择 udp,配置文件中设置 protocol UDP

2.应用的监听 IP 要设置为 vip,在这个场景中可以更改 slave DNS 的设置 listen-on port 53 {vip;物理 ip;};其中 vip 用来实现接收 lvs 的转发包并返回数据给用户，物理 ip 用来实现 master 到 slave 的同步

3.health check 需要注意，lvs 提供的 HTTP_GET 和 TCP_CHECK 都是基于 TCP 协议的（TCP 的 SYNC 包来检查应用）

对于 udp 的检查可以使用 MISC_CHECK 的方式。

比如下面个设置：

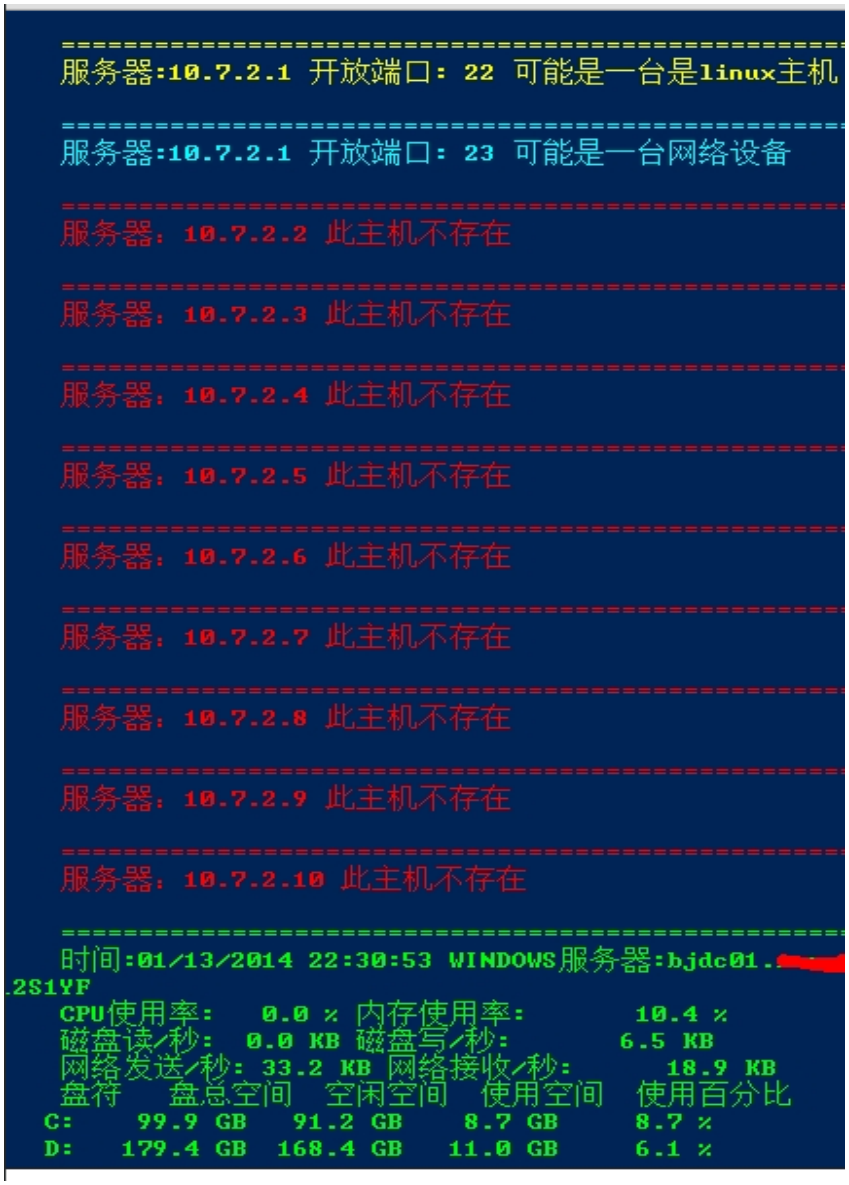
```
1 virtual_server vip 53 {
2         delay_loop 2
3        lb_algo rr
4         lb_kind DR
5         protocol UDP
6         real_server real_server1 53 {
7         weight 100
8         MISC_CHECK {
9                   misc_path "/etc/keepalived/check_named.sh real_server1"
10                    misc_timeout 5
11                }
12          }
13                real_server real_server2 53 {
```

一个可以检测网络内主机类型的脚本

作者：李小松 来源 http://lixiaosong.blog.51cto.com/705126/1351351

最近一直在写一个自动检测网络内主机类型的脚本。基本功能可以实现判断主机操作系统类型，如果是域内的主机可以获取主机的硬件参数和性能参数，并判断是否存在网络设备。对一个运维人员来说往往需要尽快熟悉一个陌生的网络。所以这个脚本就很方便了，如果有更好的建议欢迎指正感谢！

```
#Author:Lixiaosong
#Email:lixiaosong8706@gmail.com
#For:检测/24 掩码网络内主机系统类型并获取 windows 主机参数
#Version:1.0
################################################
Param(
      [Parameter(Mandatory=$true)]$Network
)
$Ip=for ($i = 1; $i -ile 255; $i += 1){"$Network.$i"}
foreach ($Ipaddress in $IP){
    #检测相关端口状态
    $Port3389=3389 | %{ echo
((new-object Net.Sockets.TcpClient).Connect("$Ipaddress",$_)) "$
true"} 2>$null
    $Port22=22 | %{ echo
((new-object Net.Sockets.TcpClient).Connect("$Ipaddress",$_)) "$
true"} 2>$null
    $Port23=23 | %{ echo
((new-object Net.Sockets.TcpClient).Connect("$Ipaddress",$_)) "$
true"} 2>$null
    $Pingtest=Test-connection -ComputerName $IPaddress  -quiet
    if ($Port3389 -like "$true"){
        #服务器信息
        $HostSN=(GWMI -ComputerName "$Ipaddress"  win32_bios).Ser
ialNUmber
        $HostFirm=(GWMI -ComputerName "$Ipaddress" win32_bios).Ma
nufacturer
        $HostModel=(GWMI -ComputerName "$Ipaddress"  Win32_Comput
erSystem).Model
        #主机信息
        $HostName=(GWMI -ComputerName "$Ipaddress"  Win32_Compute
rSystem).DNSHostName
        $DomainName=(GWMI -ComputerName "$Ipaddress" Win32_Comput
erSystem).Domain
```

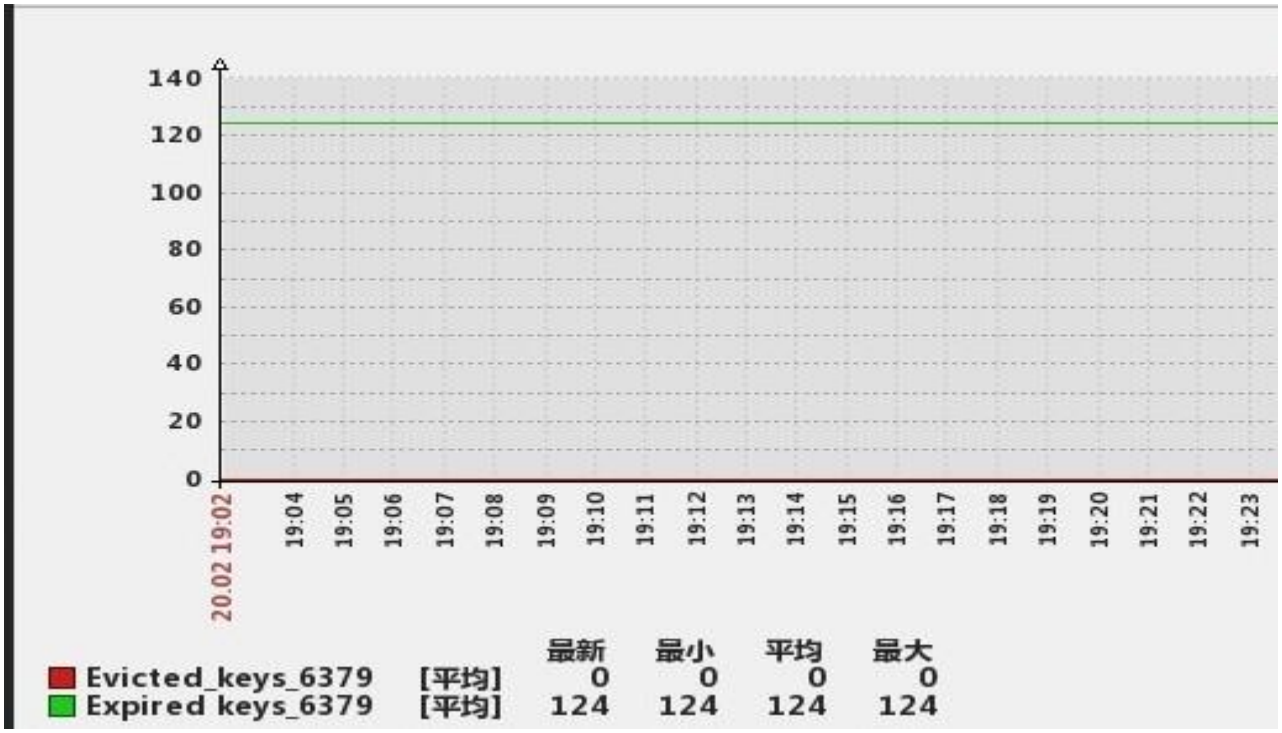

服务器:10.7.2.1 开放端口: 22 可能是一台是linux主机
服务器:10.7.2.1 开放端口: 23 可能是一台网络设备
服务器: 10.7.2.2 此主机不存在
服务器: 10.7.2.3 此主机不存在
服务器: 10.7.2.4 此主机不存在
服务器: 10.7.2.5 此主机不存在
服务器: 10.7.2.6 此主机不存在
服务器: 10.7.2.7 此主机不存在
服务器: 10.7.2.8 此主机不存在
服务器: 10.7.2.9 此主机不存在
服务器: 10.7.2.10 此主机不存在
时间:01/13/2014 22:30:53 WINDOWS服务器:bjdc01.
.2S1YF
CPU使用率: 0.0 % 内存使用率: 10.4 %
磁盘读/秒: 0.0 KB 磁盘写/秒: 6.5 KB
网络发送/秒: 33.2 KB 网络接收/秒: 18.9 KB
盘符 盘总空间 空闲空间 使用空间 使用百分比
C: 99.9 GB 91.2 GB 8.7 GB 8.7 %
D: 179.4 GB 168.4 GB 11.0 GB 6.1 %

zabbix 企业应用之固定端口监控 redis

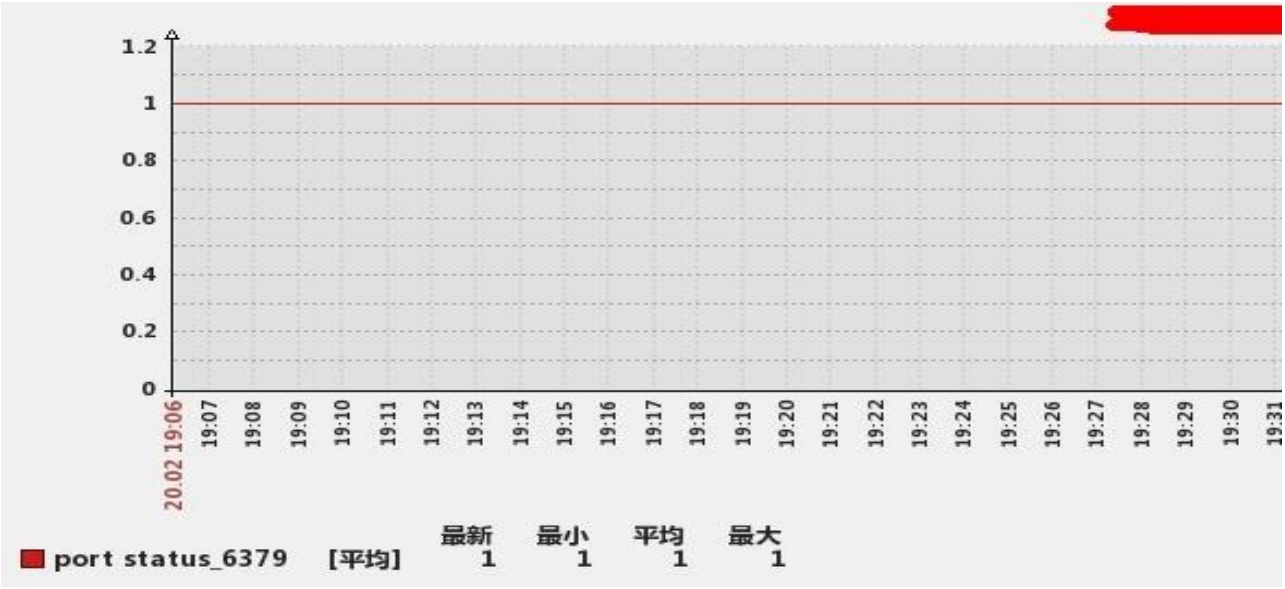
作者：邓磊 来源：http://dl528888.blog.51cto.com/2382721/1361407

本文介绍使用固定端口模式监控 redis，先展示效果图，满足你的需求在看然后监控

1、 Redis key_6379



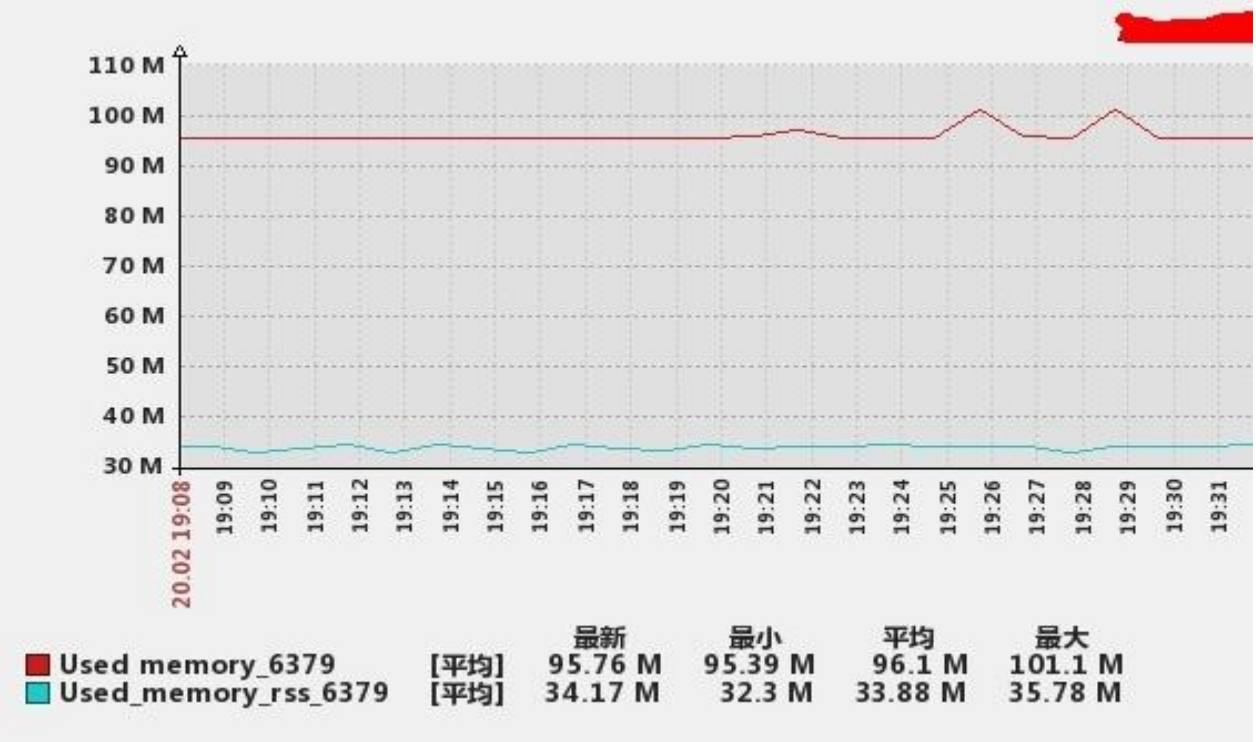
2、 Redis Last_save_time_6379



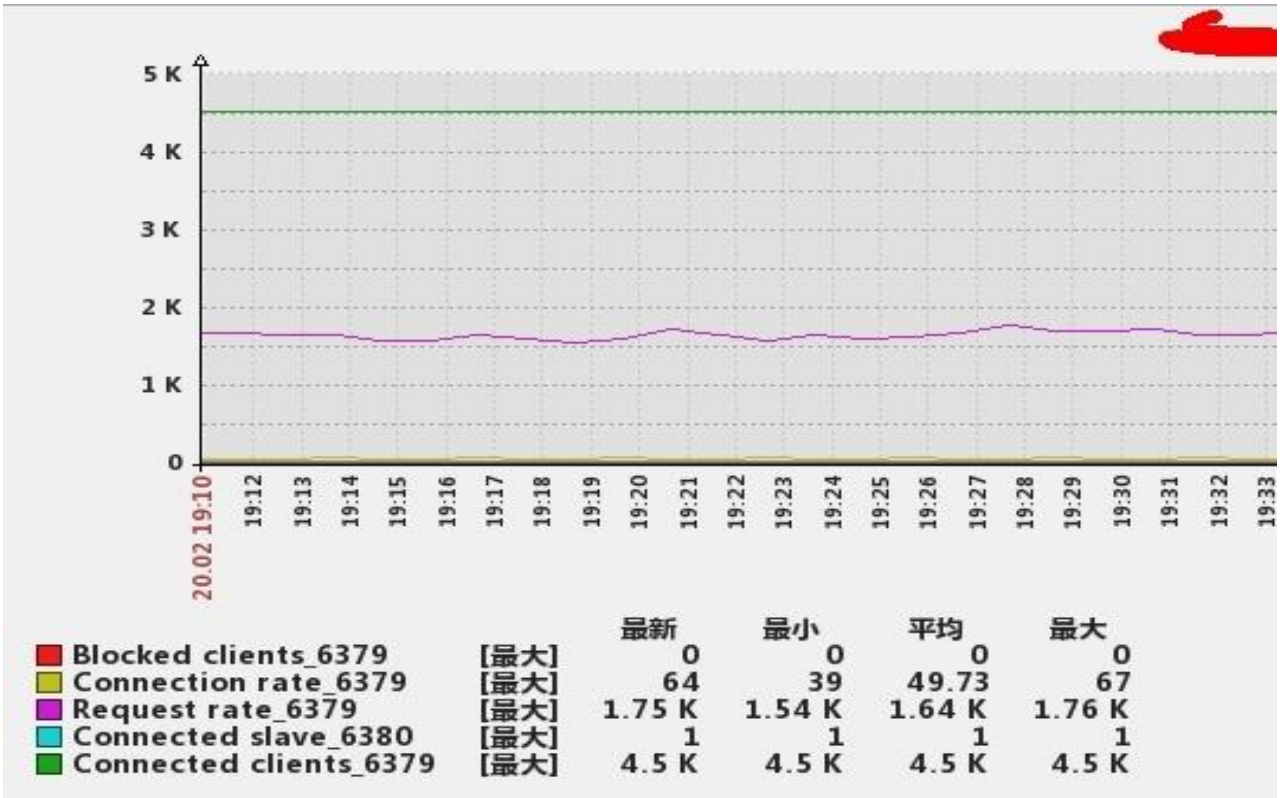
3、 Redis Port status_6379

4、 Redis Uptime_in_seconds_6379

5、 Redis Used memory_6379



6、 Redis Port Connections_6379



7、 下面是配置方案

8、 一、**在客户端**

9、 1、到/usr/loca/zabbix/conf/zabbix_agentd.conf 里添加

```
UserParameter=redis_stats[*],redis-cli -h 127.0.0.1 -p $1 info|grep $2|cut -d : -f2
```

如果你的 redis 绑定了 ip，请自行修改

2、重启 zabbix agent 服务

```
ps -ef|grep zabbix|grep -v grep|awk '{print $2}'|xargs kill -9
/usr/local/zabbix/sbin/zabbix_agentd -c /usr/local/zabbix/conf/zabbix_agentd.conf
```

二、**服务端**

1、在 zabbix 的 web 界面里连接监控 redis 模板

在 web 里选择配置-模板

然后选择导入

然后把之前下载的 zabbix_redis_6379.xml 文档导入。

然后在选择主机加入这个模板即可。

2.以上是面对 redis 端口不修改，正常为 6379 端口。

如果是多端口或者不为 6379 端口的话，可以对模板进行修改，

可以 sed -i ‘s/6379/你修改的端口/g’ zabbix_redis_6379.xml

然后在重新导入到 zabbix 就可以监控多端口或非 6379 端口

同时在 zabbix_agentd.conf 里把 UserParameter 里的 6379 改成你需要的端口

模板在附件。

发现“钓鱼网站”的一些思路

作者： 翟胜军 来源：http://zhaisj.blog.51cto.com/219066/1359824

背景：

钓鱼网站是那些模仿社交、银行、电商等网站，用于骗取用户账户名与密码的“套牌网站”。人们在访问网站时，其 URL 显示在地址栏中，但 URL 相似时(可以显示虚假 URL)，人们常常忽视，如 sohu.com 写成 s0hu.com；同时大多数情况是网站验证用户的身份，而用户不验证网站的身份，这样的结果就是用户以为自己登录的是正常网站，输入账户与密码，但实际上是把隐私信息送给了钓鱼者。

为了让用户不易发觉自己上当了，钓鱼网站有两种处理方法：一是第一次告诉你密码输入错误，让你重新输入(很多人以为自己输入手误了)，当然第二次就让你连入真实的网站了。二是自己作为中间人，帮你把账户与密码再转发给正确的网站，并把网站返回的正常信息转发给你，这样还可以监控你的实时通讯。

钓鱼网站需要你去主动访问它，所以一般采用诱惑性手段，其实方法也很简单：

1、用户输入的错误：一般是瞎猫撞上死耗子，如 sohu.com 输入成 sohu.net；

2、通过垃圾邮件发送，诱惑用户点击链接。这是钓鱼者常用的伎俩，诱人的图片、吸引眼球的新闻、感兴趣的话题、安全软件的升级包、新游戏试用……这与木马的传播有些相像；

3、入侵一些热门网站，修改用户常点击的链接，或添加一些诱惑性广告。这是黑客常用的方法，如官方网站的友好链接、电商网站的付款按钮、社交网站的常用链接…

4、在社交网站上载信息。一般是经常发布“热点消息”的人，上载热门资料，诱惑人们去点击它；

钓鱼网站的危害是不言而喻的，密码被盗的后果是严重的。无论是对用户，还是对网站都是厌恶的，钓鱼网站应该成为“风箱中的老鼠”，但如何发现它、如何识别它呢？

用户很想做的事情：

用户要识别钓鱼网站，这好像不是一件很容易的事情，既然是钓鱼网站，就做得足可以以假乱真，让你不易察觉，很多“安全专家”建议你瞪大眼睛，注意 URL 的细节，注意网页的细节…即使用户都成了“火

眼金睛”，骗术也是防不胜防，通常的结果是钓鱼网站更加“逼真”了，方法更加出乎意料了。

有两种方法是容易选择的：

1、双向认证：一些安全网站为了表明自己的身份，提供了双向认证的机制，用户可以通过网站的证书去第三方公正机关(互联网上)进行身份验证，确保自己上的是正确的网站。

但是，双向认证方法不能保护用户的利益。因为 Web 应用是访问服务器时得到的页面代码，浏览器本身没有存放“验证代码”。当访问的是正确网站时，用户端执行对服务器的认证流程；但访问的是钓鱼网站时，钓鱼者不会让用户去验证自己的钓鱼网站吧。因此，用户此时根本不知道是否执行了验证服务器的过程。

2、安全软件过滤：终端安全厂家推荐的方法，具体做法是，安装安全软件时时监控，类似防病毒软件，当用户访问某个链接时，安全软件先截获 URL 链接，并送给安全公司(安全服务商)去验证这个链接的安全性，当发现不处于“白名单”时，或者处于“黑名单”时，立即阻止用户访问，并报警。

但是，这种方法无疑增大了用户访问时的延迟，并且互联网如此广大，URL 每天都在疯狂增长，建立这样的 URL 信誉库，本身就是一件庞大的工程(目前一些互联网安全公司声称在建立)。还是有一个问题是难解决的，就是钓鱼网站本身也可以通过安全测试，进入 URL 信誉库，这样白名单策略就失灵了。

这种方法对于阻止用户访问有木马的 URL 是有用的，目前百度、Google 搜索提供对搜索结果的安全提示，就类似于这种机制。但是，对于钓鱼网站效果不是很好，因为要判断两个 URL 的页面雷同(相似有可能是钓鱼网站)，需要比对的工作量实在是太大了。

网站应该做的：

发现钓鱼网站，用户的发现是被动的，网站应该积极主动起来，不应该做旁观者。

常见的钓鱼网站有两种类型，可分别采用不同的方式去主动发现：

a)一是仿真型：模拟真实的网站；

b)二是代理型：作为中间人，代理转发用户请求与网站反馈。

代理型钓鱼网站比较好发现，因为网站会发现用户的请求来自一个互联网公共地址，它还提供用户访问页面(钓鱼页面)，你尝试访问它时，会发想同样账户与口令的请求又送回给你。

仿真型钓鱼网站比较麻烦，因为它与真实网站不联系，只是访问时的页面雷同。这好比是北京人要去查找自己的“套牌车”,除非两辆车开到了一起,才容易对比出来。首先,你如何知道自己的车已经被套牌了呢？往往是你发现有了违章，而你在那个时间就根本没有去那个地方，当警察调出你“违章”的照片证据时，你才发现那是辆套牌车。然后，你尝试去发现套牌车目前在哪里。你可以让警察调出全城甚至全国的监控录像，搜索那辆套牌车出现的位置，当然这是件几乎不可能的事情。

搜索套牌车的最佳方法是建立这样一套系统：

让全城甚至全国的道路、停车场的违章摄像头统统联网。这些摄像头都有一个共同的功能，就是可以识别出通过车辆的车牌号。你想发现车被套牌的时候(或你想看看是否被套牌时)，就把这个车牌以“通缉犯”的名义下发给所有的摄像头，当某个摄像头发现这个车牌时，立即报告自己的位置。这套系统可以与停车场的管理系统相连，停车场一般在车辆出入口都有摄像，记录进出车辆的车牌号，所以，直接搜索系统的数据库就可以得到停车场内的车辆信息了。

用这种思路去发现仿真型钓鱼网站也是可行的。我们可以先学习自己的网站，总结出容易比较的特征，如 Hash 运算。然后，去互联网上搜索其他的网站，是否可以发现同样特征的网页，若有，就应该是钓鱼网站。实际中我们比较一两个页面就可以初步判断了，不用把网站上的大部分页面都做比较。

具体的方法是：

在互联网上提供一个服务平台，输入要保护的网站 URL，学习网站页面特征(定期更新)，建立保护者单。与互联网搜索公司合作(也可以与互联网运营商合作)，因为搜索公司的爬虫、互联网运营商核心链路上流量跟踪，都是 URL 更新最快、数量最全的数据库，访问他们提供的 URL，去比较每个网页的特征是否匹配，发现雷同立即报警。

“主动发现”钓鱼网站的方案：

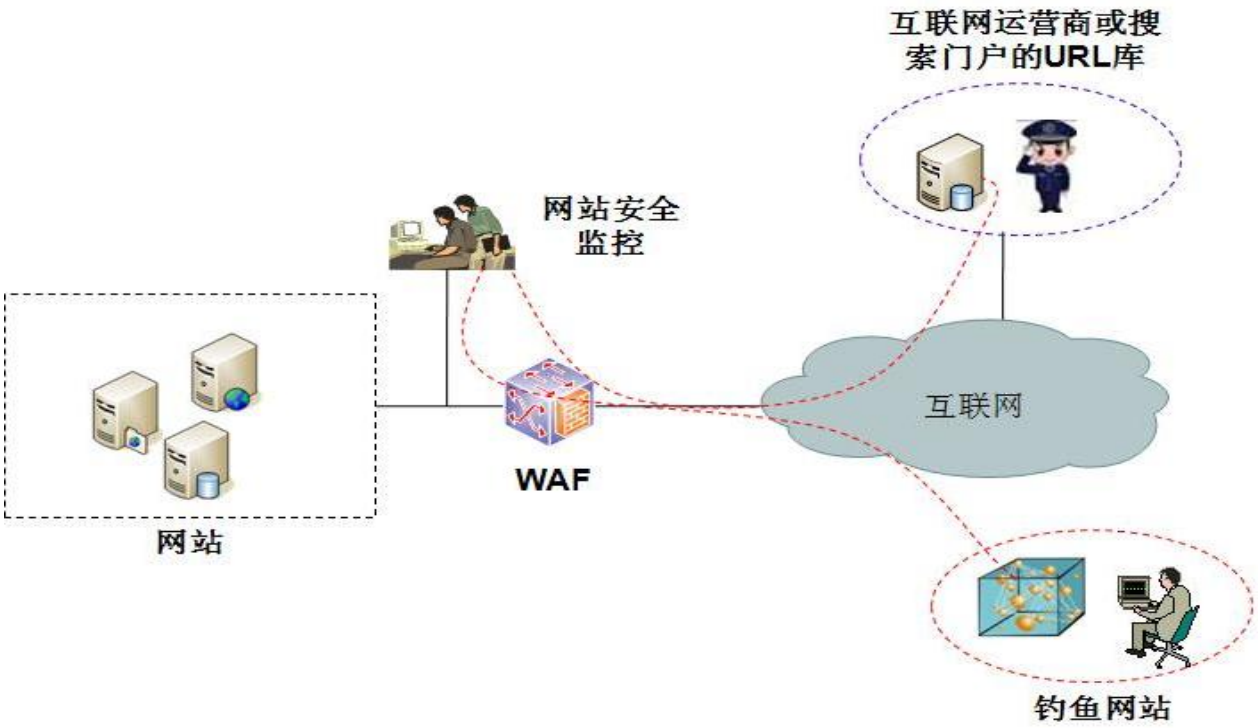
把仿真型与代理型钓鱼网站的发现思路合在一起，可以给出下面的具体方案。

网站安全监控平台可以旁路在网站出口上，主要的工作有两个：

1、与 WAF 合作，因为 WAF 是网站前的必经之路，可以提取所有访问网站者的 IP 地址，进行代理型钓鱼

网站的探测。尝试访问该地址，会发现该地址再发来一个用户请求，请求同样的页面；

2、学习自身网站关键页面(容易被钓鱼的网页)的特征，访问互联网运营商，或者是搜索门户的 URL 数据库(更新部分)，尝试访问这些页面，与关键页面的特征进行比对，发现雷同，立即报警。



该方案也可以由第三方机构建立一个互联网 URL 公共数据库，为每个网站提供钓鱼网站的查询使用。该数据库可以在互联网上部署探针，收集最新的 URL，更新数据库。

该方案还有一个好处，就是可以用来抵御 DNS 攻击或 DNS 欺骗：

建立了 URL 的公共数据库，可以同时存储 URL 的 IP 地址，当发生 DNS 欺骗时，用户可以主动进行相同 URL 的 IP 地址比对，发现不同时，可以报警。

关于 python 调用 zabbix api 接口的自动化实例 [结合 saltstack]

作者：芮峰云 来源：http://rfyiamcool.blog.51cto.com/1030776/1358792

前言：

这两天一直做一个叫集群配置管理平台的自动化项目，写了有 20 多天了，项目做的还算顺利，只是一堆的接口需要写，有点烦。因为 clusterops 项目到最后肯定是要和监控平台做结合的，这两天也抽时间看了下。 以前自己也写过不少类似 zabbix 的接口调用教程，当时看的时候，由于时间有限，也都是草草跑 demo。

zabbix的接口挺好理解 ,任何的程序都可以写 ,甚至是linux的curl命令。我这边用python的urllib、urllib2 来搞的，当然会 php 的就更好了，因为 zabbix 的接口是 php 写的，懂 php 可以直接用现成的。zabbix 官网有大量的接口，你只要会用 zabbix，然后看下 api 的说明，应该就没啥问题了

https://www.zabbix.com/documentation/1.8/api

简单说三个例子，入个门。

获取 KEY

```
!/usr/bin/env python2.7
#coding=utf-8
import json
import urllib2
# based url and required header
url = "http://monitor.example.com/api_jsonrpc.php"
header = {"Content-Type": "application/json"}
# auth user and password
data = json.dumps(
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
    "user": "Admin",
    "password": "zabbix"
},
"id": 0
})
# create request object
request = urllib2.Request(url,data)
for key in header:
```

获取 hostlist

```
#!/usr/bin/env python2.7
#coding=utf-8
import json
import urllib2
#xiaorui.cc
url = "http://10.10.10.61/api_jsonrpc.php"
header = {"Content-Type": "application/json"}
# request json
data = json.dumps(
{
    "jsonrpc":"2.0",
    "method":"host.get",
    "params":{
        "output":["hostid","name"],
        "filter":{"host":""}
    },
    "auth":"dbcd2bd8abc0f0320fffab34c6d749d3",
    "id":1,
})
# create request object
request = urllib2.Request(url,data)
for key in header:
    request.add_header(key,header[key])
# get host list
try:
    result = urllib2.urlopen(request)
except URLError as e:
    if hasattr(e, 'reason'):
        print 'We failed to reach a server.'
        print 'Reason: ', e.reason
    elif hasattr(e, 'code'):
        print 'The server could not fulfill the request.'
        print 'Error code: ', e.code
else:
    response = json.loads(result.read())
    result.close()
    print "Number Of Hosts: ", len(response['result'])
    for host in response['result']:
        print "Host ID:",host['hostid'],"Host Name:",host['name']
```

添加主机

```
#!/usr/bin/env python2.7
#coding=utf-8
import json
import urllib2
#xiaorui.cc
url = "http://10.10.10.61/api_jsonrpc.php"
header = {"Content-Type": "application/json"}
# request json
data = json.dumps(
{
    "jsonrpc":"2.0",
    "method":"host.create",
    "params":{
        "host": "10.10.10.67","interfaces":
        [{"type": 1,"main": 1,"useip": 1,"ip": "10.10.10.67","dns": "","port": "10050"}],
        "groups": [{"groupid": "2"}],"templates": [{"templateid": "10087"}]
        },
    "auth":"dbcd2bd8abc0f0320fffab34c6d749d3",
    "id":1,
}
)
# create request object
request = urllib2.Request(url,data)
for key in header:
    request.add_header(key,header[key])
# get host list
try:
    result = urllib2.urlopen(request)
except URLError as e:
    if hasattr(e, 'reason'):
```

我个人觉得 zabbix 的 rest api 难点在于 key 相关的认证 ,会了之后 ,再看官网的 api 文档就一目了然了。

啥时候用？

在我的集群平台下，我可以把暂时下线的服务器，在平台上去除，但是大家有没有想到，你要是吧主机删掉后，监控端会有一堆的通知发给你，所以，在处理主机的时候，顺便调用 zabbix 的接口，把该主机的监控项目给删掉。

在我通过 saltstack 添加 lvs 后端主机的时候，我也同样可以调用接口，把后端的主机相应的监控都给加进去。

就先这样，有时间再丰富下该文章。

JDK 在 LINUX 系统平台下的部署案例与总结

作者：孙杰 来源：http://xjsunjie.blog.51cto.com/999372/1358183

JDK

(Java Development Kit)是 Sun Microsystems 针对 Java 开发员的产品。JDK 是整个 Java 的核心，包括了 Java 的开发和运行环境，Java 工具和 Java 基础的类库。在 LINUX 系统中，进行 JDK 的正确部署是保证 JAVA 程序能够正常运行的重要前提。对于运维人员来说，还要注意在不同的 CPU 架构下 JDK 部署的差异，下面本文结合部署案例进行一下总结。

一、CPU 架构为 X86 的 JDK 部署

英特尔推出 X86 架构已满 30 年了，这种架构是大家最常见的一种架构，适用范围很广。甲骨文公司提供了这种架构的 JDK，我们可以从 ORACLE 官网下载所需版本的 JDK。

#chmod a+x jdk-6u43-linux-x64-rpm.bin

#./jdk-6u43-linux-x64-rpm.bin （路径默认在/usr/java）

这样就安装完了，设置一下环境变量

#vi /etc/profile 加入如下语句

JAVA_HOME=/usr/java/jdk1.6.0_43

CLASSPATH=.:$JAVA_HOME/jre/lib/rt.jar:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar

PATH=$PATH:$JAVA_HOME/bin

export PATH CLASSPATH JAVA_HOME

#source /etc/profile 使环境变量生效

#java -version

java version "1.6.0_43"

Java(TM) SE Runtime Environment (build 1.6.0_43-b01)

Java HotSpot(TM) 64-Bit Server VM (build 20.14-b01, mixed mode)

这样就部署完毕了，比较简单和方便。

二、cpu 架构为 power 的 JDk 部署

IBM 提供了在 cpu 架构为 power 的 linux 操作系统的 JRE 和 JDk，ORACLE 公司那里没有。如果你要在这种 cpu 架构下安装 JDk 的话，就只能乖乖的去 IBM 公司下载。获取的方法很简单，注册个账号后，找到那个 sdk 的下载地址就行。我下载的是 ibm-java-ppc64-sdk-6.0-15.0.bin，是 64 位 power。

安装很简单，简单设置一下就可以安装了

#chmod a+x ibm-java-ppc64-sdk-6.0-15.0.bin

#./ibm-java-ppc64-sdk-6.0-15.0.bin

安装过程是交互式的，需要回答一些问题，并且可以自主选择安装的路径。

最后需要配置/etc/profile：

用 VI 编辑/etc/profile，添加如下

JAVA_HOME=/home/ibm/java-ppc64-60

JRE_HOME=/home/ibm/java-ppc64-60/jre

PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin

CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$JRE_HOME/lib:

export JAVA_HOME JRE_HOME PATH CLASSPATH

重新加载 profile 文件：

source /etc/profile

然后执行一下

#java -version

java version "1.6.0"

Java(TM) SE Runtime Environment (build pxp6460sr15-20131017_01(SR15))

IBM J9 VM (build 2.4, JRE 1.6.0 IBM J9 2.4 Linux ppc64-64 jvmxp6460sr15-20131016_170922 (JIT enabled, AOT enabled)

J9VM - 20131016_170922

JIT - r9_20130920_46510ifx2

GC - GA24_Java6_SR15_20131016_1337_B170922)

JCL - 20131015_01

说明：从上面信息可以看出是 IBM JDK 64 位，小版本号：SR15。

这样就部署成功了，可以使用了。

三、cpu 架构为 PA-RISC 的 JDK 部署

HP 提供了在 cpu 架构为 pa-risc 的 linux 操作系统的 JRE 和 JDk，如果你要在这种 cpu 架构下安装 JDk 的话，就只能去 HP 公司下载。获取的方法也很简单，注册个账号后，找到那个 sdk 的下载地址就行。

我下载的是 jdk15_15000_1111.depot，是 64 位的 jdk。

将这个下载好的文件 jdk15_15000_1111.depot 拷贝到 HP 小型机上，如/tmp/目录下

安装 JDK

swreg -l depot /tmp/jdk15_15000_1111.depot

swinstall -s /tmp/jdk15_15000_1111.depot

进入安装界面：选中 java5 然后按 tab 选择 Actions -> Mark For Install，

然后选择 Actions -> Change Target，填写目标 path，选择 Action -> Install，

系统先进行分析，分析成功后点击 OK 开始安装，安装完成后选择 Done，

选择 File -> Exit 退出，JDK 就安装好了。

然后配置 JAVA 环境

将环境变量加入到 profile 中，如果写入到(/etc/profile)则对所有用户都生效，如果只对某个用户生效写入到(/home/username/profile)中即可，注意变量中=号左右不要有空格

#vi /etc/profile

添加以下代码，默认安装目录为/opt/java1.5

```
# set java environment
JAVA_HOME=/opt/java1.5
JRE_HOME=/opt/java1.5/jre
PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
CLASSPATH=.:/opt/java1.5/lib:/opt/java1.5/jre/lib:$CLASSPATH
export JAVA_HOME JRE_HOME PATH CLASSPATH
```

执行命令测试是否成功

java -version

将显示相应的 JDK 版本信息，表示 JDK 安装成功。

```
java version "1.5.0.11"
Java(TM) SE Runtime Environment (build 1.5.0.11-jinteg_20_jan_2011_05_11-b00)
Java HotSpot(TM) Server VM (build 15.3-b01-jre1.5.0.11-rc1 PA2.0 (aCC_AP), mixed mode)
```

说明：从上面信息看出是基于 HP PA 芯片的 1.5.0.11 版本的 JDK。HP JDK 的内核也是 Oracle JDK，从其官网上也能看出，如下：

Version 5.0.11 – July 2011 (includes Oracle update 5u11)

本文列举了三种不同 CPU 架构下的 JDK 的部署和环境配置，在具体应用中请大家根据不同情况进行参考配置和使用。

几个负载均衡器的小结

作者：菜菜光 来源：http://caiguangguang.blog.51cto.com/1652935/1357638

之前负责过公司 webcdn 相关的设计和维护工作 ,同时做了些应用运维方面的工作,简单谈几个我对接触过的几款负载均衡器的了解:

1.**keepalived**

使用 vrrp 实现高可用，可以理解成是 IP 层的高可用，虚拟出一个 vip 来实现高可用的功能。

实际应用时主要是两两组合，提供两个 vip，在一台挂掉后可以自动转移至另一台服务。

比较常见的问题是脑裂问题和 ip 转移后的 arp 问题 ,第一种问题可以通过 tcpdump 快速定位 rc(要抓 vrrp 协议),第二种情况可以使用 arping 的脚本解决(keepalived 的 notify_master 设置)。

基本上没有性能上的问题。

2.**lvs**

线上用的比较多的一种负载均衡器，内核态转发型的高可用工具，可以看做是 tcp 层的负载均衡器,主要原理就是对 tcp 的报文做些更改，然后进行转发,支持的状态检测方法也比较完善，有端口检测，url 检测和自定义脚本检测。

实际应用中一般是用 lvs 的 dr 模式，配合 keepalived 使用，keepalived 提供主机的高可用，lvs 提供应用的高可用，lvs 采用 url 的检测方式（避免端口 ok 但是应用挂起的情况，比如 java 应用的 gc）。

比较常见的问题是 realserver 被踢出的问题，可以通过在 lvs 部署监控并根据检测方法部署相关的检测脚本(网络检测，端口检测，url 检测)来找到问题的 rc。

很少能遇到性能的问题，在包量很大的情况下（几十万/s），可能会遇到网卡 ring buffer 和网卡中断的问题，可以通过丢包率和 mpstat 来定位，ring buffer 问题可以尝试增加网卡的 ring buffer 设置解决，中断问题可以采用 centos5.8 以上的内核配合多队列(必须)来解决。

3.**nginx**

现在主流的反向代理软件，可以通过代理的功能实现负载均衡的功能，本身功能比较完善，既可以做静态站点，缓存又可以做负载均衡器，配置上可以优化的地方很多，有端口检测和 url 的检测，但是检测的方式是被动式的，会对用户的访问造成一定的影响。

以 java 类应用来说 ,常见的使用方式是 lvs+nginx+tomcat/resin ,比较完善的日志 ,可以通过分析 nginx 日志的状态码，响应时间(response time)和后端响应时间(upstream time)来跟踪 qos，定位问题。

性能上问题不大 ,比较常见的问题是 buffer 导致的 io 问题和日志本地切割导致的 nginx 慢速响应的问题第一个问题可以尝试内存换 io 或者干脆关掉 buffer 试试，第二个问题可以把日志通过网络写入远端处理（不过貌似目前 nginx 不支持）。

4.**tengine**

nginx 的变种，在 url check 上做了改进，支持主动的检查，减少了对用户的影响，同时可以支持多种方式的日志处理(通过网络写入远端处理等)，还支持各种自定义的插件，灵活性比较高。在 webcdn 的结构设计中我就是用了 lvs+tengine+squid/trafficserver 的结构。

5.squid/trafficserver/varnish

反向代理软件，个人觉得还是做缓存会比较有优势。没有用来做过负载均衡器，这里就不妄加评论了。。

6.**haproxy**

个人感觉和 nginx 差不多，比 nginx 要轻量级。

一般都是 lvs + haproxy 来做高可用，比如淘宝的 cdn 结构就是 lvs+haproxy+trafficserver

这个个人用的也比较少。。

7.**f5 设备类**

高大上的设备，没接触过，不过有界面一切都好说。

一句话：用好一个软件比用一个好软件更重要。

写几个 Hadoop 部署用到的小脚本

作者： 向磊 来源：http://slaytanic.blog.51cto.com/2057708/1370007

最近抛弃非 ssh 连接的 hadoop 集群部署方式了，还是回到了用 ssh key 验证的方式上了。这里面就有些麻烦，每台机器都要上传公钥。恰恰我又是个很懒的人，所以写几个小脚本完成，只要在一台机器上面就可以做公钥的分发了。

首先是生成 ssh key 脚本

```
#!/bin/sh
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
```

ssh-keygen 一般来说需要输入 passphrase，但是一般都是三个回车过去了，我懒的输入，加上-P ''就不用了。

然后是添加公钥到从节点的脚本

```
#!/bin/sh
read -p "输入远端服务器 IP: " ip
ssh-copy-id -o StrictHostKeyChecking=no -i ~/.ssh/id_rsa.pub root@$ip
ssh root@$ip 'sed -i "s/^#RSAAuthentication\ yes/RSAAuthentication\ yes/g"
/etc/ssh/sshd_config'
ssh root@$ip 'sed -i "s/^#PubkeyAuthentication\ yes/PubkeyAuthentication yes/g"
/etc/ssh/sshd_config'
ssh root@$ip 'sed -i "s/^#PermitRootLogin\ yes/PermitRootLogin\ yes/g"
/etc/ssh/sshd_config'
ssh root@$ip 'service sshd restart'
hostname=`ssh root@${ip} 'hostname'`
echo "添加主机名和 IP 到本地/etc/hosts 文件中"
echo "$ip    $hostname" >> /etc/hosts
echo "远端主机主机名称为$hostname, 请查看 /etc/hosts 确保该主机名和 IP 添加到主机列表文件中"
echo "主机公钥复制完成"
```

然后是第三个脚本读取主机列表然后把/etc/hosts 复制到所有主机上。

```
#!/bin/sh
cat /etc/hosts | while read LINE
do
    ip=`echo $LINE | awk '{print $1}' | grep -v "::" | grep -v "127.0.0.1"`
    echo "Copying /etc/hosts to ${ip}"
    scp -o StrictHostKeyChecking=no /etc/hosts root@${ip}:/etc/
```

生产环境 Mysql 数据库备份脚本

作者：徐元振 来源：http://laoxu.blog.51cto.com/4120547/1353542

脚本基本功能： 徐元振

自动压缩备份 mysql 数据库

自动删除近 10 天前的备份文件

删除时显示删除进度

```
#!/bin/bash
#Author absolutely.xu@gmail.com
MAXIMUM_BACKUP_FILES=10
BACKUP_FOLDERNAME="database_backup"
DB_HOSTNAME="localhost"
DB_USERNAME="root"
DB_PASSWORD="123456"
DATABASES=(
            "openfire"
            "csp"
)
#=========
echo "Bash Database Backup Tool"
#CURRENT_DATE=$(date +%F)
CURRENT_DATE=$(date +%F)
BACKUP_FOLDER="${BACKUP_FOLDERNAME}_${CURRENT_DATE}"
mkdir $BACKUP_FOLDER
#Count the database.
count=0
while [ "x${DATABASES[count]}" != "x" ];do
    count=$(( count + 1 ))
done
echo "[+] ${count} databases will be backuped..."
# Iterate over the database list and dump (in SQL) the content of echo one.
for DATABASE in ${DATABASES[@]};do
    echo "[+] Mysql-Dumping: ${DATABASE}"
    echo -n "   Began:  ";echo $(date)
    if $(mysqldump -h ${DB_HOSTNAME} -u${DB_USERNAME} -p${DB_PASSWORD}
${DATABASE} > "${BACKUP_FOLDER}/${DATABASE}.sql");then
        echo "  Dumped successfully!"
    else
        echo "  Failed dumping this database!"
    fi
        echo -n "   Finished: ";echo $(date)
done
```

第 1 天备份数据库功能测试如下

```
[root@localhost database_backup]# for i in `seq 1 31`;do touch database_backup_2014-01-${i}.tar.bz2;done
[root@localhost database_backup]# ls
database_backup_2014-01-10.tar.bz2  database_backup_2014-01-20.tar.bz2  database_backup_2014-01-30.tar.bz2
database_backup_2014-01-11.tar.bz2  database_backup_2014-01-21.tar.bz2  database_backup_2014-01-31.tar.bz2
database_backup_2014-01-12.tar.bz2  database_backup_2014-01-22.tar.bz2  database_backup_2014-01-3.tar.bz2
database_backup_2014-01-13.tar.bz2  database_backup_2014-01-23.tar.bz2  database_backup_2014-01-4.tar.bz2
database_backup_2014-01-14.tar.bz2  database_backup_2014-01-24.tar.bz2  database_backup_2014-01-5.tar.bz2
database_backup_2014-01-15.tar.bz2  database_backup_2014-01-25.tar.bz2  database_backup_2014-01-6.tar.bz2
database_backup_2014-01-16.tar.bz2  database_backup_2014-01-26.tar.bz2  database_backup_2014-01-7.tar.bz2
database_backup_2014-01-17.tar.bz2  database_backup_2014-01-27.tar.bz2  database_backup_2014-01-8.tar.bz2
database_backup_2014-01-18.tar.bz2  database_backup_2014-01-28.tar.bz2  database_backup_2014-01-9.tar.bz2
database_backup_2014-01-19.tar.bz2  database_backup_2014-01-29.tar.bz2  database_backup.sh
database_backup_2014-01-1.tar.bz2   database_backup_2014-01-2.tar.bz2
```

10 天以后，会自动保留最近 10 天的备份，10 天以上会自动删除

模拟测试脚本功能，创建 31 天的备份文件如下

```
[root@localhost database_backup]# ./database_backup.sh
Bash Database Backup Tool
[+] 2 databases will be backuped...
[+] Mysql-Dumping: openfire
        Began:  Wed Jan 22 18:34:26 PST 2014
        Dumped successfully!
        Finished: Wed Jan 22 18:34:26 PST 2014
[+] Mysql-Dumping: csp
        Began:  Wed Jan 22 18:34:26 PST 2014
        Dumped successfully!
        Finished: Wed Jan 22 18:34:26 PST 2014

[+] Packaging and compressing the backup folder...
database_backup_2014-01-22/
database_backup_2014-01-22/openfire.sql
database_backup_2014-01-22/csp.sql

[+] There are 1 backup files actually.
[root@localhost database_backup]# ls
database_backup_2014-01-22.tar.bz2  database_backup.sh
```

每天计划任务定时执行脚本保留最近 10 天的

```
[root@localhost database_backup]# ./database_backup.sh
Bash Database Backup Tool
[+] 2 databases will be backuped...
[+] Mysql-Dumping: openfire
        Began:  Wed Jan 22 19:22:01 PST 2014
        Dumped successfully!
        Finished: Wed Jan 22 19:22:01 PST 2014
[+] Mysql-Dumping: csp
        Began:  Wed Jan 22 19:22:01 PST 2014
        Dumped successfully!
        Finished: Wed Jan 22 19:22:01 PST 2014

[+] Packaging and compressing the backup folder...
database_backup_2014-01-22/
database_backup_2014-01-22/openfire.sql
database_backup_2014-01-22/csp.sql

[+] There are 31 backup files actually.
[+] Remove 21 old backup files.
[+] Safeting the newest backup files and removing old files...
Removing:[##########################################################]100%
[root@localhost database_backup]# ls
database_backup_2014-01-22.tar.bz2  database_backup_2014-01-26.tar.bz2  database_backup_2014-01-30.tar.bz2
database_backup_2014-01-23.tar.bz2  database_backup_2014-01-27.tar.bz2  database_backup_2014-01-31.tar.bz2
database_backup_2014-01-24.tar.bz2  database_backup_2014-01-28.tar.bz2  database_backup.sh
database_backup_2014-01-25.tar.bz2  database_backup_2014-01-29.tar.bz2
```

备份

记录一次奇葩的 sleep(15)引起的 Too many connections

作者：贺春旸 来源：http://hcymysql.blog.51cto.com/5223301/1362788

上周日早上快到 6 点，天还没亮，睡得正香，突然被一声短信惊醒，Too many connections！

Mysql 10.8.8.163	check_mysql_connections	CRITICAL	2014-02-23 05:56:27	hechunyang	service-notify-by-sendemail
Mysql 10.8.8.163	check_mysql_connections	CRITICAL	2014-02-23 05:56:26	sa	service-notify-by-sendemail

我这纳闷呢，搞啥活动呢？平常这都是低风期啊。

登陆机器一查看慢日志，发现有这么一条 SQL，很奇葩：

```
# Time: 140223  6:11:57
# User@Host: dedeseek[dedeseek] @  [10.8.8.165]
# Query_time: 2913.759786  Lock_time: 0.000072 Rows_sent:
use gfan_dede;
SET timestamp=1393107117;
select down from dede_rom where aid=1-sLeEp(15);
# User@Host: dedeseek[dedeseek] @  [10.8.8.165]
# Query_time: 3104.683401  Lock_time: 0.000084 Rows_sent:
SET timestamp=1393107117;
select down from dede_rom where aid=1-sLeEp(15);
# User@Host: dedeseek[dedeseek] @  [10.8.8.165]
# Query_time: 3485.051838  Lock_time: 0.000045 Rows_sent:
SET timestamp=1393107117;
select down from dede_rom where aid=1-sLeEp(15);
# User@Host: dedeseek[dedeseek] @  [10.8.8.165]
# Query_time: 3124.984049  Lock_time: 0.000044 Rows_sent:
SET timestamp=1393107117;
```

跑了快 1 小时了，而且查询的结果还是 0 行。

很奇怪到底在搞毛？我又查看了这个表，并没有 aid=1 的记录。

```
mysql> select aid from dede_rom order by aid limit 10;
+-------+
| aid   |
+-------+
| 26244 |
| 26247 |
| 26248 |
| 26249 |
| 26250 |
| 26251 |
| 26252 |
| 26253 |
| 26254 |
| 26255 |
+-------+
10 rows in set (0.00 sec)
```

周一询问了开发，都不知道这条 SQL。最后猜想到，这是不是监控人员，搞的一个每隔 15 秒就连接一下，看看数据库存活状态？假如是的话，应该用程序做，直接 1-sleep(15)很危险，假如我这个表里有 100 行，那么耗时就是 100*15 秒，这样会一直占用连接不释放，结果造成连接堆积，正常的连接进不来，导致业务报错。

最后直接 pt-kill，结束了这场 Too many connections 风波。

自动化运维平台中的统一认证接入与单点登录实现

作者：芮峰云 来源：http://rfyiamcool.blog.51cto.com/1030776/1362424

前言：

在工作中，大大小小开发了不少自动化运维平台，能更好的提高效率以及人工的失误。有朋友问我，登录平台的账号密码如何的管理。当听到这个问题的时候，我说直接入库呀，但是说完后，觉得相当的不妥，最少和我现在解决方案也不一样。

以前做运维开发项目的时候，每个 app 都是一套用户密码，顶多做了一个加密流转，申请接入的时候，需要领导的邮件审批，后期改成 ldap 在 openldap 做认证了。 但是现在不这么搞了。

原因呢？ 我相信大家公司肯定都有认证的接口，先说下常见的认证接口有那两种（我就见过这两种，希望朋友补充）：

第一种，公司的各个平台通过 ldap 连入 windows ad 目录，或者是连入特定的 db。 当时这样很不安全，如果我是开发者的话，用户输入用户名和密码的时候，点击登录，我完全可以把账号密码给 print 出来。

第二种，就是 Passport 的方案，算是统一认证，一般是 oauth，但是 oauth 相对麻烦点。大家有时候，要登录论坛的时候，他会提示可以用 qq，支付宝，人人的账号登录。 我现在的方案就是类似这种的方法，好！ 下面说下，我对这个统一认证的接口使用。

我自己也写过一个 统一认证平台接口，是基于 cookie 方式的，实现的方法相对简单，但是很有效，安全方面让安全部门测试过，在一定条件下，还是很安全的。在以前公司开发的多个平台也都接入了这个统一认证。

语言：python

框架：tornado

模块：requests，flask-admin，oauth2

对于 python oauth2 的方案，可以看 https://github.com/simplegeo/python-oauth2

```
得到授权码 code
获取 access token
通过 access token, 获取 OpenID
通过 access token 及 OpenID 调用 API, 获取用户授权信息
```

这里的方案不是 oauth2，因为 oauth2 更多的是给大批的第三方应用设计的，咱们这里只是做 passport 认证，用 oauth2 显得不是太合适。

原文：http://rfyiamcool.blog.51cto.com/1030776/1362424

集群平台 clusterops clusterops.xiaorui.cc，中心认证 passport.xiaorui.cc 为例，

1. 判断用户是否登录，已经在 passport 登录的话，可以直接访问 clusterops，如果没有登录的话，会转跳到 passprot.xiaorui.cc/redirect ? urlto=clusterops.xiaorui.cc

2. 当转跳到 passprot.xiaorui.cc/redirect ? urlto=clusterops.xiaorui.cc 的时候，会提示用你的域

账户和密码登录，登录成功后，会再次转跳到 clusterops.xiaorui.cc/api?res=aqggzwnasdzo9kzwsxedclmcksduwe8sdf0d&Token=F3fQk1eTJWu2XbWHEzuXXJ0KoJeH6O

3. clusterops 接收 api 接口的 res 和 Token 字段数据，用这两个字段以 cookie 的方式去访问 passport.xiaorui.cc/getuserdata?accesstime=时间戳

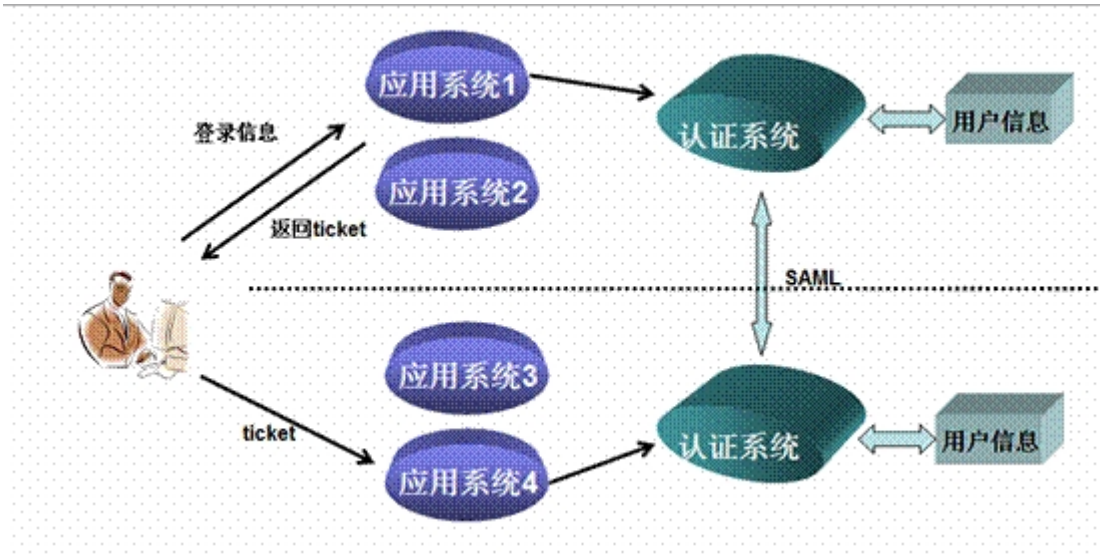
4. 统一认证端 passport 拿到了 cookie 在 redis 里面做对比，然后判断 access 时间，在 5 分钟之内，符合要求给他 return 用户数据，如果不符合，就给他回一个错误数据。

5. clusterops 解析了返回的 json 数据，把 username 及一些数据，放到 session 里面，这样用户就完成了认证及登录。

6. 退出的话，可以选择该 app，也可以选择清楚 sso 的单点登录的 sesson 标记。

原文：http://rfyiamcool.blog.51cto.com/1030776/1362424

看这个流程图比较直观点，一般来说我们只是对于 web 应用做认证接入，认证后，在这个 web 应用的所有动作权限 ,都是在 web 应用本身做的。如果每个应用和连接都再次向认证系统去验证 cookie、session，那就会有瓶颈，所以用 saml 使两个认证服务器做认证通信 ！



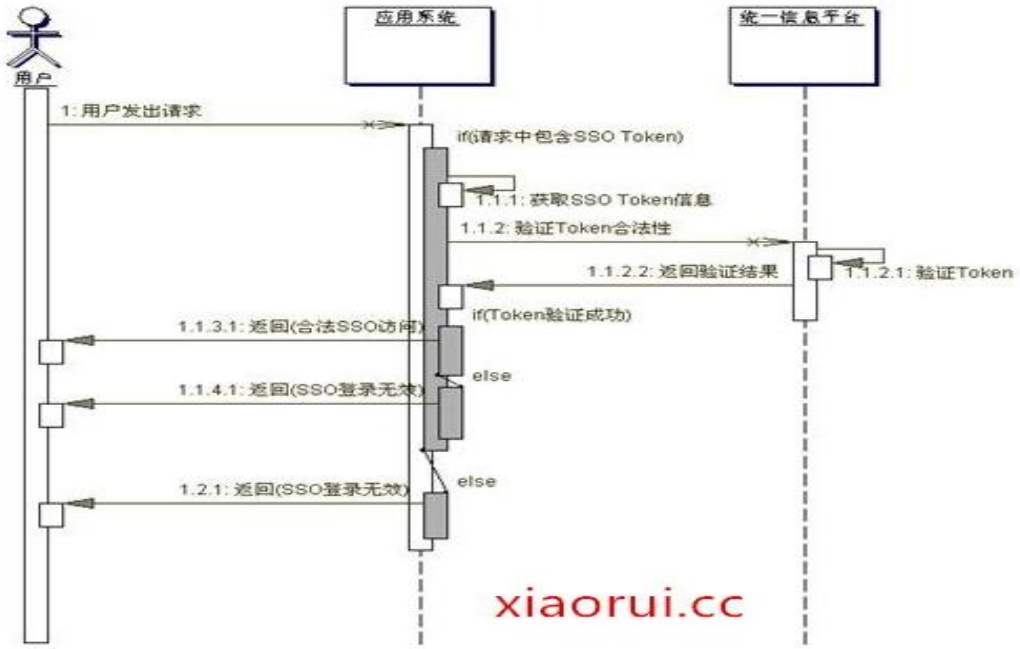
上面说的只是统一认证方面的，下面再来说说单点登录

单点登录 SSO（Single Sign-On）是身份管理中的一部分。SSO 的一种较为通俗的定义是：SSO 是指访问同一服务器不同应用中的受保护资源的同一用户，只需要登录一次，即通过一个应用中的安全验证后，再访问其他应用中的受保护资源时，不再需要重新登录验证。

开源的 sso，也有不少，我用过 lemonldap 这个 websso 方案，感觉还不错哈。

http://lemonldap-ng.org/documentation

SSO 的好处：



方便用户：从用户实际使用角度考虑

用户使用应用系统时，能够一次登录，多次使用。用户不再需要每次输入用户名称和用户密码，也不需要牢记多套用户名称和用户密码。

方便管理员：从日常维护管理角度考虑

系统管理员只需要维护一套统一的用户账号，方便、简单。相比之下，系统管理员以前需要管理很多套的用户账号。每一个应用系统就有一套用户账号，不仅给管理上带来不方便，而且，也容易出现管理漏洞，开发者也不知道用户的账号密码

简化应用系统开发：从应用扩展角度考虑

开发新的应用系统时，可以直接使用单点登录平台的用户认证服务，简化开发流程。单点登录平台通过提供统一的认证平台，实现单点登录。因此，应用系统并不需要开发用户认证程序。

单点登录在这里就很好体现了，只要用户登录了 passport 统一认证，登录后平台会 session 标记，只要别的项目指向 PassPort 认证，那自然而然他们都可以登录了。

捎带角说下，我们一定要再开发一套针对统一认证 passport 的权限控制，不然每个应用都自己控制 admin 和 guest 的权限，这样该又乱套啦！ 这段时间我也搞了一套权限管理接口，可以配套在统一认证接口上，权限控制的很细，一个页面，一个菜单，一个动作都可以控制。

欧了，这里是我自己对于运维平台的统计认证，sso 单点登录，权限管理的理解。 我相信，我的这篇文章更加适合新手理解，当然肯定也有理解出问题的地方，请大神指正 ！

DG 主库 fial over，强制激活备库解决案例

作者：wyan117 来源：http://2874575.blog.51cto.com/2864575/1360166

14 年 1 月份的时候，因硬件环境的变更，需要把库从原来的存储平台移到新的存储平台。也就是把数据库的底层存储介质更换一下。下面主要记录一下事故的发生及应对措施。

事情概况

win 平台，11R2，64 位，单实例，DG 物理备库。主库与备库均只有 redo 和业务数据文件存储介质为 fusion io 卡，其它数据文件、控制文件等是存放在非 fusion io 卡介质上的。现需要将存储介质 fusion io 卡更换为 virident 卡。2 种卡都是直接插在 pci 插槽上的，且两卡本身是由同一生产厂家生产过了。最主要是有人反应之前测试用时直接用文件拷贝方式来进行移库的。鉴于 2 种卡的拷贝速度很快，于是采用了直接数据文件卡对卡拷贝方式来进行换卡操作，然后将盘符改成原来一模一样的，卡的拷贝速度很快，忘了是几百 M 每秒，而且只拷贝存放在 fusion io 卡上的数据文件到 virident 卡上，其它的数据文件及控制文件等 东西不需要移动。

沦入事故

于是乎，先将各服务都关掉，主备库都 shutdown。开始针对主库采取文件卡到卡的对拷。很快拷贝完了，盘符也改好了。于是启动主库，启动时报错如下：

```
Errors in file E:\APP\ADMINISTRATOR\diag\rdbms\branch_p\branch\trace\branch_lgwr_2196.trc:
ORA-00313: ??????? 3 (???? 1) ???
ORA-00312: ???? 3 ?? 1: 'J:\REDO\BRANCH\REDO03.LOG'
ORA-01382: ?? 1 ???????? J:\REDO\BRANCH\REDO03.LOG????????? (4096) ???????? (512)
Errors in file E:\APP\ADMINISTRATOR\diag\rdbms\branch_p\branch\trace\branch_lgwr_2196.trc:
ORA-00313: ??????? 3 (???? 1) ???
ORA-00312: ???? 3 ?? 1: 'J:\REDO\BRANCH\REDO03.LOG'
ORA-01382: ?? 1 ???????? J:\REDO\BRANCH\REDO03.LOG????????? (4096) ???????? (512)
```

虽然有乱码，但是从报错错误代码可以看出是无法打开日志组。经查发当无法打开的日志组为 current。从 ora-01382 可知，是因为日志文件的 block size 比 virident 卡的磁盘扇区大小要大。所以日志组无法打开。开始没注意到 ora-01382 这个错误，心想先想办法把库给拉起来，于是使用命令 alter database clear unarchived logfile group 3;

可是报错依旧如上。后来找到解决 ORA-01382 这个错误了，用 alter system set "_disk_sector_size_override"=true; 更改参数解决启动报错无法打开日志的问题，然后继续启动数据库，结果能正常启动，但是库依旧执行前面的 clear log 操作，这一 clear 了就会使得数据有丢失，于是就赶紧 cancel。然后把拷贝到 virident 卡上面的数据全部清空，准备重新将 fusion io 卡上的数据拷贝过来。然后在库 startup mount 后设置隐含参数 _disk_sector_size_override 值再 open 库。

已经将 virident 卡上的数据清掉了，再次将原数据文件拷贝过来了，正拷贝过程中想到，这时的控制文件已经在之前 clear log 时 SCN 发生了变化。fusion io 卡上的数据文件头 scn 已经与此时控制文件的 scn 不一致了，数据库是无法打开了。之前 clear log 期间发生的日志切换（也就是生成的 archivelog ）也都清了，这样归档日志文件也不连续了。备库与主库之前的日志出现 gap 了。

事故解决

现在有 2 种解决方法。在拷贝文件之前做了一个 rman 备份，此备份是在停掉业务时，准备换卡前做的备份，所以用它数据绝不会挂失，我们可以用 rman 备份文件来做恢复。另一种方法是换卡前已确保发生了业务的数据全部传入备库，所以可以把备库拉起来当主库用。此时的 DG 环境是主库 fail over 了，备库东西都完好。

定下一步的解决方案。

鉴于 rman 备份文件恢复起来速度过慢。而我们距离正常业务开展时间少于 3 个小时。然后决定采取将备库拉起来当主库用这种方法。但是备库一样要把 fusion io 卡上的数据移到 virident 卡上，还是采取文件对拷方式。省时。

这次在拷贝前（库已停）静态备份所有的数据文件，以防再次发生在主库上面的错误。备份完后，将 fusion io 卡上的数据物理拷贝到 virident 卡上，并把盘符互换，接下来库 startup mount ， alter system set "_disk_sector_size_override"=true; 再 open read only。可以正常打开。好了，存储介质更换成功了。下一步是要把备库激活当主库用，是强制激活备库，关库启动到 mount，使用命令 alter database activate standby database; 打开了备库，再次重启一下备库。连上业务系统，能正常使用。于是再关机，拔下 fusion io 止。到了正常业务操作时间，此库已经可以当生产库来用了。原来的主库就废掉，重新建一个 physical standby。

不要随随便便的 distinct 和 order by

作者：qdjyyl 来源：http://qdjalone.blog.51cto.com/1222376/1360112

有客户反应网站后台订单相关查询非常慢,通过程序拿到了相关 sql。

explain

```
explain SELECT DISTINCT(o.orders_id), o.oa_order_id, customers_email_address, o.order_type, ot.text AS total_value, o.track_number, o.date_purchased, o.orders_status, o.specialOperate, o.isSpecialParent, o.pay_ip, o.supply_id, o.products_center_id, o.split_code, o.is_import, o.shipDays,o.delivery_country,o.use_coupon ,o.payment_method FROM orders AS o LEFT JOIN orders_total AS ot ON ot.orders_id=o.orders_id AND ot.class='ot_total' WHERE 1 AND o.is_delete = 0 AND o.date_purchased >= '2013-09-30 10:00:00' AND (o.specialOperate = 0 OR o.isSpecialParent=1) ORDER BY date_purchased DESC, orders_id DESC LIMIT 0, 20;
+----+-------------+-------+-------+---------------------------------+-----------
---------------+---------+---------------------+--------+-----------------------
----------------------+
| id | select_type | table | type  | possible_keys                   |
key                           | key_len | ref                 | rows   |
Extra                                       |
+----+-------------+-------+-------+---------------------------------+-----------
---------------+---------+---------------------+--------+-----------------------
----------------------+
|  1 | SIMPLE      | o     | range | date_purchased                  |
date_purchased                | 9       | NULL                | 606632 | Using where;
Using temporary; Using filesort |
|  1 | SIMPLE      | ot    | ref   | idx_orders_total_orders_id,class |
idx_orders_total_orders_id | 4       | banggood.o.orders_id
|     19 |                                             |
+----+-------------+-------+-------+---------------------------------+-----------
---------------+---------+---------------------+--------+-----------------------
----------------------+
2 rows in set (0.05 sec)
```

使用 profiling 发现 Copying to tmp table on disk 占用了大部分性能。

仔细查看该语句并和开发讨论 ,发现 distinct 和 ORDER BY date_purchased DESC, orders_id DESC 中，distinct 关键字可以省略,而且 ORDER BY date_purchased DESC, orders_id DESC 可以去掉后面的 orders_id desc(开发对多个字段排序不理解).

去掉后，再次 explain.

```
mysql> EXPLAIN
    -> SELECT o.orders_id, o.oa_order_id, customers_email_address, o.order_type, ot.text AS tota
    -> ORDER BY date_purchased DESC LIMIT 0, 20;
+----+-------------+-------+-------+---------------------------------+-------------------------
| id | select_type | table | type  | possible_keys                   | key
+----+-------------+-------+-------+---------------------------------+-------------------------
|  1 | SIMPLE      | o     | range | date_purchased                  | date_purchased
|  1 | SIMPLE      | ot    | ref   | idx_orders_total_orders_id,class | idx_orders_total_orders_
+----+-------------+-------+-------+---------------------------------+-------------------------
2 rows in set (0.01 sec)
```

索引使用情况不变,但是下面的 profiling,发现结果瞬间出来,执行时间不过 0.003s,而且已经没有了 Copying to tmp table on disk 状态。

总结：1.因为 distinct 关键字需要对结果集进行去重,如果天然无重复,是不需要加上去重关键字的,上面的例子结果集有将近百万,去重字段又多，在 tmp_table_size 以及 sort_buffer_size 中排序已经不够用,所以将结果集复制到磁盘，严重影响速度

2. order by a,b 开发人员很喜欢用类似的语句，尽管对功能没有多大作用

为 Angularjs ngOptions 加上 index 解决方案

作者 :格茸扎西 来源 :http://whitewolfblog.blog.51cto.com/3973901/1359470

今天在 Angularjs 交流群中有位童学问道如何为 Angular select 的 ngOptions 像 Angularjs 的 ngRepeat 一样加上一个索引$index.

其实对于这个问题来说 Angular 本身并未提供$index 之类的变量供使用。但是也不是说对于这个问题我们就没有解决方案。

把这个问题换成角度来看,我们所需要的就是 js 数组的下标 ,所以我们如果我们能够在对象上加入下标，使用表达式作为 option 的 label 就能解决了。

但是第一印象让我想起的是 js 数组本来就是一个 key/value 的对象，只是 key 为数组下标而已，所以有了如下之设计：

html:

```
<pre></pre><select  ng-model="a" ng-options="value.field as getDesc1(key,value) for (key,value) in t"></select>
```

js:

```
$scope.getDesc1 = function(key, value) {
    return (parseInt(key, 10) + 1) + "->" + value.field;
};
```

可是不幸的是如果对于 JavaScript 你若将他作为 key/value 对象那么 key 将是无序的所以，出现了无序的下标如下：

```
<select ng-model="a" ng-options="l.field as getDesc1(key,value) for (key,value) in t
" class="ng-valid ng-dirty">
  <option value="0" >1-&gt;jw_companyTalent</option>
  <option value="1" >2-&gt;jw_reportgroup</option>
  <option value="10" >11-&gt;jw_ads</option>
  <option value="11" >12-&gt;jw_jobcomment</option>
  <option value="12" >13-&gt;jw_companyInfo</option>
  ....
</select>
```

所以这样是无法解决的。还好博主还有一招，ngOptions 支持 Angularjs 的 filter，所以我们可以对数据源对象上加上一个 order 字段标示下标作为序号。并且你可以在一个 2 年前的 Angular 的 issue 中看到 Angular 已经 fix issue，option 会对数组进行按下标顺序生成。

html:

```
<pre></pre>
<select  ng-model="b" ng-options="l.field as getDesc(l) for l in t | index "></select>
```

js:

```
var app = angular.module('plunker', []);
   app.controller('MainCtrl', function($scope) {
     $scope.t = [{
       "field": "jw_companyTalent"
     }, {
       "field": "jw_reportgroup"
     }];
     $scope.getDesc = function(l) {
       return l.order + "->" + l.field;
     };
   }).filter("index", [
     function() {
       return function(array) {
         return (array || []).map(function(item, index) {
           item.order = index + 1;
           return item;
         });
       };
     }
   ]);
```

这下 option 是按照有序的生成，最后我们终于能完美解决了,所以本文也将收尾。在最后在附上可运行的 demo plnkr ngOptions index;

轻量级性能测试工具之 Apache Benchmark

作者：李小鹏 来源：http://starpoint.blog.51cto.com/968349/1358793

AB（apache benchmark） 为 Apache 自带的性能测试工具在 APACHE 的 bin 目录下。
通过 CMD 进入 apache 的 bin 目录下，本例以 windows 下的 apache 为例。
C:\Program Files\Apache Software Foundation\Apache2.2\bin>
格式： ab [options] [http://]hostname[:port]/path
参数：

-n requests Number of requests to perform
//在测试会话中所执行的请求个数。默认时，仅执行一个请求
-c concurrency Number of multiple requests to make
//一次产生的请求个数。默认是一次一个。
-t timelimit Seconds to max. wait for responses
//测试所进行的最大秒数。其内部隐含值是-n 50000。它可以使对服务器的测试限制在一个固定的总时间以内。默认时，没有时间限制。
-p postfile File containing data to POST
//包含了需要 POST 的数据的文件.
-T content-type Content-type header for POSTing
//POST 数据所使用的 Content-type 头信息。
-v verbosity How much troubleshooting info to print
//设置显示信息的详细程度 - 4 或更大值会显示头信息， 3 或更大值可以显示响应代码(404, 200 等), 2 或更大值可以显示警告和其他信息。 -V 显示版本号并退出。
-w Print out results in HTML tables
//以 HTML 表的格式输出结果。默认时，它是白色背景的两列宽度的一张表。
-i Use HEAD instead of GET
// 执行 HEAD 请求，而不是 GET。
-x attributes String to insert as table attributes
-y attributes String to insert as tr attributes
-z attributes String to insert as td or th attributes
-C attribute Add cookie, eg. 'Apache=1234. (repeatable)
//-C cookie-name=value 对请求附加一个 Cookie:行。 其典型形式是 name=value 的一个参数对。此参数可以重复。
-H attribute Add Arbitrary header line, eg. 'Accept-Encoding: gzip'
Inserted after all normal header lines. (repeatable)
-A attribute Add Basic WWW Authentication, the attributes
are a colon separated username and password.
-P attribute Add Basic Proxy Authentication, the attributes

are a colon separated username and password.

//-P proxy-auth-username:password 对一个中转代理提供 BASIC 认证信任。用户名和密码由一个:隔开，并以 base64 编码形式发送。无论服务器是否需要(即，是否发送了 401 认证需求代码)，此字符串都会被发送。

-X proxy:port Proxyserver and port number to use
-V Print version number and exit
-k Use HTTP KeepAlive feature
-d Do not show percentiles served table.
-S Do not show confidence estimators and warnings.
-g filename Output collected data to gnuplot format file.
-e filename Output CSV file with percentages served
-h Display usage information (this message)

//-attributes 设置 属性的字符串. 缺陷程序中有各种静态声明的固定长度的缓冲区。另外，对命令行参数、服务器的响应头和其他外部输入的解析也很简单，这可能会有不良后果。它没有完整地实现 HTTP/1.x; 仅接受某些'预想'的响应格式。 strstr(3)的频繁使用可能会带来性能问题，即，你可能是在测试 ab 而不是服务器的性能。

参数很多,一般我们用 -c 和 -n 参数就可以了. 例如:

ab -c 1000 -n 1000 http://127.0.0.1/index.php

【linux 下为

./ab -c 1000 -n 1000 http://127.0.0.1/index.php】

这个表示同时处理 1000 个请求并运行 1000 次 login.jsp 文件.

ab -c 10 -n 10 http://10.14.132.35:8080/cs/user/login.jsp

window 下为

C:\Program Files\Apache Software Foundation\Apache2.2\bin>ab -c 100 -n 100http://192.168.1.175:8080/cs/user/login.jsp

【linux 下命令为：

#/usr/local/apache2/bin/ab -c 1000 -n 1000 http://192.168.1.175:8080/cs/user/login.jsp】

This is ApacheBench, Version 2.0.41-dev <$Revision: 1.121.2.12 $> apache-2.0

Copyright (c) 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/

Copyright (c) 1998-2002 The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.1.175 (be patient).....done

Server Software: Apache-Coyote/1.1

//平台 apache 版本

Server Hostname: 192.168.1.175

//服务器主机名

Server Port: 8080

//服务器端口

Document Path: /cs/user/login.jsp
//测试的页面文档
Document Length: 7536 bytes
//文档大小
Concurrency Level: 100
//并发数
Time taken for tests: 13.125 seconds
//整个测试持续的时间
Complete requests: 100
//完成的请求数量
Failed requests: 0
//失败的请求数量
Write errors: 0
//写的错误数量
Total transferred: 788000 bytes
//整个场景中的网络传输量
HTML transferred: 753600 bytes
//整个场景中的 HTML 内容传输量
Requests per second: 7.62 [#/sec] (mean)
//大家最关心的指标之一，相当于 LR 中的 每秒事务数 ，后面括号中的 mean 表示这是一个平均值
Time per request: 13125.000 [ms] (mean)
//大家最关心的指标之二，相当于 LR 中的 平均事务响应时间 ，后面括号中的 mean 表示这是一个平均值
Time per request: 131.250 [ms] (mean, across all concu
//每个请求实际运行时间的平均值
Transfer rate: 58.63 [Kbytes/sec] received
//平均每秒网络上的流量，可以帮助排除是否存在网络流量过大导致响应时间延长的问题

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	0	102 886.7	16	8875
Processing:	188	9357 1702.7	9906	10156
Waiting:	16	9300 1954.0	9906	10156
Total:	188	9458 1562.2	9922	11281

//网络上消耗的时间的分解
Percentage of the requests served within a certain time (ms)
50% 9922
66% 10016
75% 10047

80% 10063
 90% 10125
 95% 10156
 98% 10156
 99% 11281
100% 11281 (longest request)
//整个场景中所有请求的响应情况。在场景中每个请求都有一个响应时间，其中 50%的用户响应时间小于 9922 毫秒，最大的响应时间小于 11281 毫秒

一次解决 DB2 接口文件到 Oracle 无法导入问题的经历

作者：王保强 来源：http://baoqiangwang.blog.51cto.com/1554549/1357891

前几天触点营销平台出了点问题，请同事帮忙处理，结果两天过去了，还是没定位到问题。
临近春节，还是要把问题解决掉的，今天忙碌了一上午总算解决这个问题
从 DB2 主机往 Oracle 主机发送接口文件，接口文件的数据总是缺失一部分，导致 CRM 系统无法看到相应的营销活动。

于是你说我没传接口文件，我说你没法处理接口文件，其实也很容易处理，关键是一个扯字，呵呵，看下接口文件是否在服务器上以及是否与两端的数据库一致即可。
好不容易厘清了接口文件问题，又扯到接口文件本身内容的问题，从接口机上看到的文件部分内容是乱码，且存在回车换行问题。
又开始了新一轮的测试，其实两端都应该测试接口文件能否正常入库，在 DB2 上测试数据的导出和导入都是正常的，那接口主机上看到的乱码是什么情况呢？估计数行字节太长，导致无法全部展示，所以系统把部分中文字符给分割了，导致看到的文件内容是乱码。
在 DB2 上的测试很简单，导出数据，加载数据，展示数据即可。

```
export to u:/IW3001test0001.AVL of del modified by coldel0x01 nochardel striplzeros decplusblank
select * from hnwangbq.yingxiaoan;
create table hnwangbq.test
(
sale_act_id varchar(20),
sale_act_name varchar(50),
act_begin_date varchar(20),
act_end_date varchar(20),
data_time varchar(20),
sale_act_script varchar(500),
sms_script varchar(500),
sale_act_type varchar(20)
);
import from u:/IW3001test0001.AVL of del modified by coldel0x01 nochardel decplusblank
insert into hnwangbq.test;
select * from hnwangbq.test;
```

在 DB2 上处理的一切都很正常，接下来测试在 Oracle 上的导入。
没办法只得重新安装了一下 Oracle 和 PL/SQL Developer，创建新表。

```
create table hnwangbq.test
(
sale_act_id varchar2(20),
sale_act_name varchar2(50),
```

```
act_begin_date varchar2(20),
act_end_date varchar2(20),
data_time varchar2(20),
sale_act_script varchar2(500),
sms_script varchar2(500),
sale_act_type varchar2(20)
);
```

通过 PL/SQL Developer 的 Text importer 进行导入测试，发现报错是字段行太短，接下来继续查看发现 Oracle 把几条记录给合并到一条记录中了

为什么出现这种情况呢，发现行中存在"双引号情况，Oracle 把"双引号作为列的 Quote character，所以把两个"双引号中的内容作为一个字段了

于是通过 replace 函数把"双引号改为'单引号，再次测试，搞定

不过在生产环境的测试又出现问题了，还是出现错误，经过阅读 shell 脚本发现 Oracle 是通过 SQLLDR 进行接口文件加载的，我也通过 SQLLDR 进行了测试，控制文件如下：

```
LOAD DATA
INFILE 'd:\IW3001test0001.AVL'
REPLACE INTO TABLE test
FIELDS TERMINATED BY X'01'   OPTIONALLY ENCLOSED BY '"'
trailing nullcols
(SALE_ACT_ID     ,
SALE_ACT_NAME    ,
ACT_BEGIN_DATE ,
ACT_END_DATE     ,
DATA_TIME        ,
SALE_ACT_SCRIPT ,
SMS_SCRIPT       ,
SALE_ACT_TYPE
)
```

发现也是报错误，错误的原因在加载日志中，错误信息如下：

```
记录 1: 被拒绝 - 表 TEST 的列 SALE_ACT_SCRIPT 出现错误。
数据文件的字段超出最大长度
记录 3: 被拒绝 - 表 TEST 的列 SMS_SCRIPT 出现错误。
数据文件的字段超出最大长度
记录 8: 被拒绝 - 表 TEST 的列 SMS_SCRIPT 出现错误。
数据文件的字段超出最大长度
记录 11: 被拒绝 - 表 TEST 的列 SMS_SCRIPT 出现错误。
。。。
记录 42: 被拒绝 - 表 TEST 的列 SMS_SCRIPT 出现错误。
数据文件的字段超出最大长度
```

表 TEST:
27 行 加载成功。
由于数据错误, 15 行 没有加载。
由于所有 WHEN 子句失败, 0 行 没有加载。
由于所有字段都为空的, 0 行 没有加载。

通过百度搜索了一下，发现其他人也有同样的问题，问题的原因在 Oracle 的 SQLLDR 在缺省的情况下对字符串的处理是 CHAR(255)，而部分字段的列显然超过了 255 个字节，于是修改了一下控制文件。

```
LOAD DATA
INFILE 'd:\IW3001test0001.AVL'
REPLACE INTO TABLE test
FIELDS TERMINATED BY X'01'   OPTIONALLY ENCLOSED BY '"'
trailing nullcols
(SALE_ACT_ID     ,
...
SALE_ACT_SCRIPT char(2000),
SMS_SCRIPT     char(2000),
```

其实解决方案不外乎两个，一个方法是在数据源侧上进行控制，一个方法是在加载侧进行不停的测试和完善。

问题的解决之道无它，无非是测试+测试而已，有时候看似解决了一个问题，另外一个问题又出来了，问题的根源在于不停的探索。

其实探索也是一种乐趣。

redis 多实例重启脚本

作者：王伟 来源：http://wangwei007.blog.51cto.com/68019/1351628

redis 属于单进程的服务，它主要受内存、CPU、磁盘 IO（主要是做持久化），如果服务器配置比较高，多核 CPU、高内存的服务器，可以考虑做 redis 多实例。做多实例之前，首先要考虑 CPU 和内存的利用，我在测试的时候发现，redis 在 QPS 为 6-8W 左右的时候，这个 redis 所在的逻辑 CPU 核的负载就在 100%左右，所以要优化 CPU 使用这块，目前一般是是做网卡软中断来实现平衡这种单进程使用 CPU 过高的情况，不过需要网卡支持网卡软中断，效果不错。

多实例 redis 的管理，涉及到监控、服务的管理等，这里只说 redis 多实例的重启。以下是多实例的重启脚本，仅供交流参考：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import redis,threading,sys,socket,time,os
from multiprocessing import Pool
def port_check(host,port):
    POOL = redis.ConnectionPool(host=host, port=port)
    my_server = redis.Redis(connection_pool=POOL)
    try:
        a = my_server.ping()
        if a:
            return 1
        else:
            return 0
    except Exception,e:
        return 0
def shutdown_server(host,port):
    flag = port_check(host,port)
    if flag == 1:
        POOL = redis.ConnectionPool(host=host, port=port)
        my_server = redis.Redis(connection_pool=POOL)
        my_server.shutdown()
        while 1:
            flag = port_check(host,port)
            if flag == 0:
                print "%s:%s is Shutdown OK" % (host,port)
                break
            else:
                time.sleep(1)
    else:
        print "%s:%s is Shutdown alreaday" % (host,port)
def start_server(host,port):
```

装腔，你不能不学

作者：孙继斌 来源：http://sunjibin.blog.51cto.com/918334/1360843

这半年，见到了两位老同事。当然不是只见到这两位老同事，而是这两位让我印象极为深刻。

一位多年未见。竟然杀入了影视圈，投资拍电影。请我抽古巴雪茄、法国红酒、入住广电大厦。言谈间，不喜欢去附近餐馆，因为电视台台长什么的，熟人太多。

我觉得他很了不起。

一位多年未见。移民某国多年，据说是某国华商协会的副会长，现在回国来找机会。他说：每次回来时都是司长亲自去机场接我。他说：在某国，我从事跨国并购，跑过上百个国家和地区。他还说：我在 500 强做副总裁。问他：你怎么当上的商会副会长？他说：我请克林顿来协会演讲，之后地位立刻不一样了。半年后公推我当副会长。

我深深觉得~

最后，我终于在微信订阅了几个装腔信号。内容题目经常是这样的：《第一次坐飞机，怎样装作经常坐的样子？》《怎样在微博扮上流社会》《如何在华尔街装土豪》……

今年我读到这样一句话：如果你把你想要得到的，当做你已经得到的，那么有一天你会发现你准定会得到。这句话很有内涵。

记住你想要什么，这是我去年所发现的最重要的工具。去年夏天参加一次聚会，身旁的小圣女说：许多人都认为我很了不起，但我不觉得自己如何了不起。我最大特点是，我的回答与众不同。当别人请我聚餐，我不想出份子钱，我会说我正一个人在法国吃海鲜；当别人问我现在在哪里工作，尽管我在一个不知名公司，我会说我在 500 强就职；当别人问我的职位，尽管我是一个普通文员，我会说我是业内知名总监；当别人问我 PPT 做的怎么样，尽管我很少做 PPT，我会说和秋叶水平有的一拼……。当然，你知道，不过一两年，这些都变成了现实。

请记住你想要什么，然后将它表现出来。不要怕别人说你不自量力、装腔作势、无知无畏。现在这个时代，似乎是一个疯狂的时代。英特尔公司前总裁安迪-格鲁夫说：只有偏执者才能生存。现如今，意义似乎有了迁移。原来强调的是基于过去经验的偏执，现在则是强调基于未来想象的偏执。这种偏执，看来可以帮助你实现你的梦想。因为几乎所有的事情，所有的成就、所有的尊严、所有的经验，在社会的快速变化面前都烟消云散了，只留下真正重要的事情：想象。记住现在的你很快就要失去，能够使你避免陷入许多心理误区。据我所知，这是最好的办法了。

过去一切的意义都在迅速衰减。原来和别人谈起《中国人的紧箍咒》，我会说是一部不错的文化著作，关于儒家文明的一家之见。现在，我会说：这是一部开创性的文化著作，有人说它和《史记》《论语》一样伟大，有人说这是百年来最好的中华文明史。这样说，其实只是为了找到过去，呼唤未来。

扬长避短，这句话千古不易。其实我们这里正在讲“扬长”。在新的时代，我们必须要学会扬长。社会的迅速变迁，正在让我们的长处迅速变短。这是时间的力量。时间的流速似乎在不断加快。我们必须走在时间的前面，长处也是如此。扬长已经来不及了，我们必须先扬出去，再将长处补长。

我们的时间是有限的，因此不要浪费时间等待别人的品评。乔布斯说：不要被教条所羁绊，这样你就是在根据别人思考的结果来生活。不要让其他人的观点所发出的声音淹没了你自己内心的声音。最重要的是要有勇气听从你自己的心灵和直觉。它们总会知道你真正想成为什么人，其他一切事情都是次要的。

做你将来想做的，说你将来想说的，或者有人说你装，随他们吧~你已经是赤条条无牵挂了，没有理由不听

舍本求末的运维自动化技术热潮

作者： 曹亚孟 来源：http://caoyameng.blog.51cto.com/4975863/1359732

运维自动化是 2010 年开始炒得很热的一个概念，也让很多工程师、用人单位瞎激动了很久，我也跟风学过 puppet 和 python，求职双方也经常在面试时花大量时间谈运维自动化。

但冷静下来想想，所谓自动化，只是让培训机构赚钱的噱头而已。

一句话概括运维自动化

单说“运维自动化”几个字太抽象容易被主观塞进去很多概念，上百科搜索到 IT 运维自动化的介绍又太详细、大帽子太多。

如果把运维自动化在一句话说清楚，比较官派的说法就是：“运维自动化就是在企业业务越来越复杂、对 IT 人员要求越来越高……balabalabla……的前提下，靠人工已经无法满足运维工作的需求，只能靠自动化技术来解决这一问题。”

如果用比较粗糙的说法就是“活多人少的情况下，运维不想靠堆人力去解决繁琐的问题，只能靠运维自动化来给自己减负。”

运维自动化理论与现实相悖

粗看这些理论挺有道理，但仔细分析根本不是这么回事。首先，我们真的忙了吗？

我认为运维的工作量并没有随着企业需求越来越复杂而变大，就算变大也不是靠自动化能解决的体力活。

运维自动化是给运维用的，请各位运维想想，我们的日常工作，这些年来有太大变化吗？

初级运维大部分时间在做上线和监控，高级运维在改结构修 bug。对于那些重复性的工作，云计算供应商能比你做的更好，云主机、云监控、云 RDS、云存储等等服务都是在给运维减负。

企业业务需求越来越复杂是真的，具体来说是技术进步企业要求越来越刁钻了。数据库要求主从实时同步，存储不能用 NFS 要用分布式，前端业务要求无缝切换等等。我是不是谈偏题了，这些东西跟运维自动化有什么关系？你意识到问题就好，我们这些年新增的业务需求，没多少是可以靠运维自动化解决的，要解决这些问题，还要靠我们自己的脑子。

运维自动化=shell 脚本

其实我们一直在做运维自动化，因为我们会用 shell 脚本。

我们可以说只要企业需求有变动，我们就要搭服务、搭监控，做这些事情都要写自动化脚本。当你激动的说到“自动化脚本”的时候，我想问一下，你不会写 shell 脚本吗？

搭完某个服务以后，一个有经验有责任心的运维，自然会写好系统优化脚本，复制监控监控模版。如果我们用 puppet，用 python，最后一样脱不了指定主机名的工作。

我们完全可以用 shell 脚本完成各种模拟运维操作的动作 熟练使用 shell 脚本也是每个运维的必修课，我们有必要为了一个噱头去学习 python 吗？

我曾经看过 puppet 的官方文档，他能管理的资源列出来的有“文件” “属主属组” “挂载” “软件包” “服务” “-exec 使用本地 shell” ，在我看来其实也就是“文件”和“-exec” 。

在 linux shell 脚本里，关于运维有这么多命令“cp、scp、nc、ssh、rsync、svn、chmod、chown、service、/etc/init.d/” ，这些命令已经够用了

我用 puppet 的时候，只是用他频繁监控几个重要的系统配置文件。上线的工作真正繁琐在要把 realserver 从 LB 上摘下来，且需要用人力去判断能不能摘。

具体摘设备、传新备老代码、重启 java 容器、回滚代码的工作我都写好脚本了，就这样还因为麻痹大意而出了几次高负载或丢步骤的情况。

如果能运维自动化的东西，必然能写 shell 脚本搞定，如果用 shell 脚本搞不定的东西，“运维自动——挂” 。

运维自动化≠优化

老生常谈，运维应该眼界高一些，不要总是忙着优化手头的工作，而要想手头的工作有没有必要。

有朋友肯定要说我的工作不到家，上线居然还需要人力判断，我承认这是问题，但这问题在架构不在运维。如果上线不需要人工干预，为什么不直接让开发执行？甚至更进一步，让应用服务器定期去 svn 上检测有没有新代码？在测试环境我们也会用 hudson 和 maven 让开发自己搞 ,但我肯定做好一个系统镜像保证他们把系统玩坏了也能快速恢复。

在生产环境里，运维该做的不应该是纠结一步人肉操作该用 shell 还是 python 代劳，而是说好好去推动一下，能不能多上几台服务器，能不能降低一下耦合度，不要让我们手动盯着上线工作了。

我现在的公司后台做的不好，很多业务相关的 sql 修改都要开发写好语句给运维执行。如果这个时候我给 mysql 安装个 phpadmin 就是本末倒置，写个脚本能自动传 sql 过去还是本末倒置，我实际该做的是催促公司尽快做出来企业管理后台可以让运营和客服人员直接去改业务数据。

我们写多少牛逼的 python 脚本，不如做一个稳定到单机房断电都不会宕机的架构；用好运维自动化很牛逼吗？是的，就跟用好某种文本编辑器一样牛逼。

运维自动化背后的利益推动

鼓吹自动化的大师里，很多位其实是运维开发两条腿都很短的杂鱼。

我曾经看到过一个运维自动化的教程，作者很认真的教我们，如何用某种自动化工具调用本地 shell，用 sed 命令将 crontab 里的 ntpdata 任务时间给变更了。看到这一段，我被他的执着蠢哭了，所谓的自动化居然是用 ntpdate 更新系统时间。

我也见过某大师写的自动化代码，朋友告诉我他的 python 水平只值 6k——连异常都不处理，我用半瓶醋的水平仔细看了一下他的源码我真的笑出来了，每隔几行必然能看到一个 os.system(“shell 命令”)。

在工作环境里，我用“tar /var/aaa/bbb/ccc/*.jpg”这类通配符匹配出来目标文件，写了个 10 行的脚本，将某高手用 perl 写了 100 多行，但其实就是 find+tar 的脚本给替换掉了。

在处理数据的时候，我也写 python 脚本，因为效率远超 shell 脚本。但运维自动化一定要用 python 脚本，更新文件必用 puppet，对高手来说这是风格，对新手来说这是跟风。

有心的朋友可以帮忙查一下，从 2010 年开始，都有哪些培训机构新增了运维自动化课程或 python 运维课程，又有哪些人靠这些技术把自己包装成了大师。

运维自动化的困境

那些高端大气上档次的运维自动化教师们，永远无法回避我这两个问题：

1、在一个 100 台机器下的小公司，搞运维自动化是不是在自己立项冒功？你写好的运维自动化系统，是不是配合着把文档写的很细很好了，会不会系统升级一下就运维自动挂了。

越是小公司，越容易出现单台机器跑多个业务、不同机器的环境变量完全不同的情况。假设你是个技术新兵，不用自动化只会挂一台机器，用自动化挂一堆机器；假设你是个技术高手，你知道其中的风险更不会盲目的信任一个脚本。

2、在 500 台机器机器以上的大公司，确实很需要运维自动化，否则光是手动画网络拓扑图和加监控就能累死人。

但在这个环境里，最重要最有含金量的是系统架构的设计和演进；运维自动化只是减负的工作而已，哪有聪明人放着金砖不要却要板砖的？

你觉得有没有可能这个公司几十个技术高手天天为上传个 js 文件累的要死，就等你一个空降兵来部署自动化系统解救他们的？

做运维自动化，必然是自己公司内部的服务器有大量增加，增加到你觉得手动操作很累的地步，这个时候做运维自动化是水到渠成的。但运维自动化的工作一般是企业内部已有的运维来推动的，这不应该当作招人的理由。

运维自动化也不是简单的写一些脚本或部署文件同步工具，它没有真正成型的方案，因为这是要用机器模拟运维工程师的劳动方式。每个运维团队的工作风格都不同，生搬硬套外来的自动化方案只会让我们邯郸学步举步维艰。

如果你入职第一个月就被要求设计部署自动化方案，那只能证明这个公司确实没有运维人才，且这个公司很闲其实不需要运维。

重新审视运维自动化

运维自动化的目的，放低端点，就是解决运维手动操作容易出错的问题，放高端点是希望运维忽略具体命令而更重视最终成果。

在低端领域，我们可以很自信的说，用 shell 脚本就是运维自动化；在高端领域，肯定已经搭建好了自动化环境供我们观摩学习和修改；如果你有幸参与到大规模自动化部署，那是确实是一次很有趣的挑战；在一个更高的层次上，你会发现诸如 系统标准化、应用模块化、统一认证系统等等更有价值但没人炒作的技术。运维自动化不用专门去学习，自动化的“大师”也不用刻意招聘。

最顺手的工具就是最好的工具

IT 人热爱某个技术就应该成为某项技术的主人而非信徒 ;我在文中多次强调 shell 脚本的可用性是因为 shell 脚本是每个运维必须掌握的技能。

在本文中我大量引用了对时下最热门的几个自动化运维工具的一些批评案例。但这些样例并不是用来攻击这些技术本身。事实上 puppet 应用 QQ 群是我唯一持续加入的 Linux 技术群 ,而我明白自己的 Python 水平看复杂的代码都费力。只因为这两个技术被野风吹的最火，我用他们来说明每杆大旗下都少不了盲从的人。

如果你坚信某个技术是特别强悍的并对我的言论怒火中烧，请你想想你能用你的工具做到的事情，在我的环境里能不能提前绕过，就算碰到了我能不能用 shell 脚本解决掉。我并不反对你推广你的方案，但我认为“循环调用 SSH 命令是一个我能接受的、可行的方案”。

我们应该减少盲从，拿起最顺手的工具去做一番事业，而不是玩赏最精美的道具却迷失了目标。

我比我的领导差在哪

作者：z00w00 来源：http://z00w00.blog.51cto.com/515114/1355175

正文前的话：

说实话，想写这篇文章我半年前甚至更早就想写了。但是一直没有写，原因有很多。首先客观原因是最近我很忙，忙的手忙脚乱的。主观原因是虽然有了主题，我在考虑用什么样的形式来呈现给大家。因为经常看到很多朋友在抱怨自己的领导多么多么差。就在昨天技术部“尾牙”聚餐桌上，一个比较有威信的同事说了一番这样的话：“我发现我们公司有一个特别有意思的现象。很多公司的员工都抱怨自己的领导多么多么差，而我们公司这种言论很少。我不想分析这种情况，因为毕竟离主题太远。以下我不得不以我部门的领导和我作为原型进行多方位的比较，但我想强调一点的是，文中的我，并不能代表就是我，文中的领导也不一定就完全对应我的领导。以下文章采购对比和评论的方式进行

一、年龄对比

领导的年龄：30 而立

我的年龄：20 出头

评论：从年龄上看，我们两者之间差了近 10 岁。很多人的心态就怕的就是大龄员工遇到年轻领导，这个时候让人很不爽。30 多岁的人少了一些稚嫩与激情，但是多的确实成熟和稳重。经常有年纪大的人会这样说，我吃过的盐比你吃过的米还多呢。通常会遭到年轻人的鄙视。他们的观点是：年纪大一些，就开始倚老卖老了。实际上倚老可能会有一点，但是卖老我看也未必。就想没恋爱过的人，不知道恋爱史什么滋味，没生过孩子的母亲不知道当母亲的艰辛。年纪大一些，生活阅历和社会经验毕竟丰富一些，考虑的问题就多一些

二、学历对比

领导的学历：在读硕士研究生毕业

我的学历：大专

评论：很多人对学历的看法是这样的，都是应届毕业生闯江湖的的砝码，对于很多老江湖就无所谓了。但是在我的眼里并不是这样，最起码很多企业的人力资源并不是这样看的。学历是很多企业的门面，试想一个企业的高管（或者项目经理）如果只有初中文化程度的话，怎么拿的出手给客户作为谈资。所以我用

一个较悬殊的差异来做对比，希望大家能正视，低没有关系，但是如果觉得学历低还无动于衷的话，那么就是你的不是了。

三、从业经历对比

领导的从业经历：某岗位或领域从业 5 年以上

我的从业经历：虽然工作经验 5 年以上，但是相对岗位领域从业 2 年以上

评论：因为频繁的跳槽，导致在不同的岗位上都有过从业经验，但是在和领导垂直领域的工作经验偏少。（比如领导为运维经理，我为运维工程师）工作经验能不能拿来当资本。我个人认为是可以的。排除掉各种奇葩的个例，从业一年就会经历很多与之相关的事情，从接收、处理到结果都是经验的积累。当然如果你遇到一个整日里清闲的无事可做的岗位自当别论。

四、某公司入职时间

领导在某公司工龄：5 年以上

我的某公司工龄：2 年以上

评论：很多人都比较讨厌在公司谈资历，一说到在公司多少多少年就像吃老本一样。实际上，我们每一天都在经验和体验很多事情。除了工作，我们在某个公司每天理解和体验着公司的种种制度、文化。比如有的公司管理很严格，那么你就老老实实的，有的管理很宽松，当然你就自在一些。还有你的领导的领导，你领导的领导的领导的做事风格。如果不能很好的把握，那么即使你认为做的很少，也不会得到领导的赏识。因为你必须知道领导心里最在乎的是什么。

五、技术技能

领导的技术技能：杂而全，全而不精

我的技术技能：精而不杂，杂而不全，全而不深

评论：我个人是比较讨厌外行领导内行的。所以对于一个 IT 岗位的领导，完全不懂技术是不行的。这样他就没有办法判断你做的事是否可行，也就没有办法进行决策。如果是我遇到一个外行领导的话，那只能寻机走人了。忽悠领导是一件很愚蠢的事。一旦忽悠领导造成重要事故，大部分公司的做法是领导降职，

员工辞退。因你的错误造成你的领导与你的两败俱伤，得不偿失。我的领导会画思维导图，但我不会。我的领导可以做出整年的部门预算，但是我及时能做，也不一定满足需求。我的领导比我更熟悉公司有多少台用于生产的机器，都是做什么用的，我不清楚。但我可以熟练的定位一般系统故障，并能快速的解决故障问题。如果我与领导同台 PK 故障排除。那么被淘汰的一定是他。因为我整天面对的就是这个，但是他不是。但是估计永远不会有这样的比赛。

六、技术决策

领导的决策：站在部门的角度

我的决策：站在个人岗位的角度

评论：牛顿说过：“我之所以看的很高，是因为我站在巨人的肩膀上“所以很多人都会说高度决定一切。在很多的时候，我们做事情都会遇到各种决策。我做事事情有一个习惯，就是在做一些比较重大的事情一定会和领导汇报（这个肯定的）但是我会在心中默想一个领导可能会做出的决策。然后通过领导最后真实的决策来验证我的决策。通过多次对比，我个人认为大部分与我意见向左的决策领导的决策正好，更正确，最关键的是能得到上级领导的认可。而只有少部分决策，我的决策是更合理的。这也就可能是我无法撼动他地位的最重要的原因吧

七、业务、流程的熟悉度

我的领导：较高

我：较低

评论：一个技术人员是否需要了解和掌握业务方面的知识呢？答案是肯定的。如果你的领导是空降过来的，那么熟悉业务流程必定需要一个过程。如果是一步一步走上来的，那么对于你而言，他肯定比你更熟悉公司的业务和流程。当你遇到一个需要多部门协调沟通才能解决的事的时候，那么首先的问题是什么？没错，就是找谁沟通。如果领导在这方面比你更熟悉，也就意味着比你多一项资源。不过还好，随着时间的推移，他的这项资源有一部分必定被你共享。

八、应急响应

我的领导：高

我：较高

评论：俗话说”拿人钱财替人销灾“从雇佣与被雇佣的关系来看，你拿着企业发给你的钱，就要给企业办事。当生产系统发生故障的时候，很多技术、运维甚至开发就需要有一个快速应急响应的故障。通过一段时间的观察，我发现我的领导，我的领导的领导经常大半夜起来回邮件、有一些故障（非系统报警短信）总是第一时间收到，然后第二时间通知我。当遇到一些比较紧急的事情，领导必须要快速的拿出处理办法，而我们只需要根据处理办法具体的去做就可以了。假如你一个领导与员工换位一天的机会，当遇到这种紧急情况，能不能处理，甚至说处理的是否得当，我还真不好说。

九、性格与品行

我的领导：内向型、较懒惰、务实性、沉稳

我：内向型、也较懒惰、务实性、但浮躁、易怒

评论：我和我的领导在性格和品行上还比较相似，也就是所谓的物以类聚吧，从整体上来看，这也可能是我和我的领导相处比较融洽的一个主要原因吧。我的领导比较懒，只要大方向给你指出了，剩下的事你自由发挥，不出圈就行。而且幸运的是，他很务实，干的好了就给大家争取利益，做的不好就当面指责，从不留情。比起那些当面称兄道弟，背后捅刀子的人，那真是没法比。年轻人浮躁是很正常的，被领导骂也是正常。你的领导上面还有领导。当你和领导因为问题争的面红耳赤的时候，可以想一想他是如何和上级领导沟通的呢。

十、跨部门沟通

我的领导：强

我：较弱

评论：我不知道有多少人会遇到和跨部门领导沟通的情况。特别是跨部门的领导比你的领导级别还高而需要沟通的时候。都说同行是冤家，其实同公司不同部门也是冤家。因为沟通毕竟是要解决问题，特别是需要部门联合解决的问题。人类的普遍心态就是把责任和义务都推出去，把权利和利益都留下来。面对高高在上的跨部门领导，做到不卑不亢容易，做到能争取到部门的最大利益而且让跨部门领导满意双赢就难

了。特别是书面沟通，通过邮件之类的形势，要特别小心。因为一个不留神，就有可能被跨部门领导抓住小辫子，打闷棍、穿小鞋就稀松平常了。所以很多时候我最担心的就是这方面造成的问题。而经过领导修改后的书面信息，能最大程度既保护部门利益，也不会让跨部门领导拍案而起，甚至有些涵养不高的领导，问候你的先人。

最后我想起《中国合伙人》中黄晓明饰演的“新梦想”老总在被众人一顿辱骂和扔东西的情况下，还高声的喊道“中国人就会窝里斗，去和比你强的人去学习，去学习他身上的优点”

领导就是领导，无论你心里怎么不平衡。你在这个企业一天，就得在工作中服从他的指挥。当你看到他满身缺点的时候，你试图把他想成你，看看“你”身上有什么优点。放心，只要你认真去找，一定有的。

技术人创业至今的反思

作者：毕成功 来源：http://passover.blog.51cto.com/2431658/1354968

出来创业已经 2 年时间，至此还没有任何起色，处在选择是否死的分叉口上。回顾过去，从美好憧憬走到残忍现实，一路上有太多的教训，不论以后如何，现在做一个总结给我过去的 2 年一个交代。如果以后翻身了，这叫失败前的摸索，如果真的这么死掉，就是又一个死在创业路上的菜鸟。

【选对方向，猪都能上天】

创业做重要的就是要选对方向，这就是事半功倍的关键。这个方向要根据市场情况、个人情况等众多因素决定。我觉得手游的大方向真的没任何问题，但是我个人之前没这块背景，当时没去游戏公司卧底一下，这真的是失败的一个关键原因，太多时间放到无谓的摸索上了，还迟迟找不到一个业内的人士入伙。

【产品为核心，市场为关键，技术为基础】

创业之后发现，技术真的只是成功的一个必要保证而已，只是说能让产品做出来，但是仅仅充当产品是否成功的必要条件。看到几个创业团队产品选择的很好，虽然东西做的很一般，但是最后还是得到了很不错的估值。再反观自己的项目选择，看起来做的还不错，但是收益都相当不理想，这就是差异。

创业之后深刻感觉到选择项目的重要性，花一个月选择一个正确的项目比不停做两个失败的项目更重要。当然这还牵扯另外一个问题，谁能知道什么项目是正确的？我咨询了一些人，而后发现大家都有同样的疑问，谁要是知道具体做什么一定挣钱，那都赚钱了。所以也有人提出了快速试错的方法。但是我觉得我试错的次数太多、时间太久，这应该归结到自己的方向感上有问题了吧。

其实产品和技术的定位在创业之前都还算有比较客观的认识，但是市场这块是之前完全没足够重视，或者说是没有正确的认识。现在才知道真是渠道为王的时代，有些团队就针对渠道去做项目，这真的是聪明的办法。说白了，创业成功的必要过程就是“选择项目、做出项目、卖掉项目”，而最后卖掉才是最终目的。我们团队还是有着不错的执行力，选择的项目都能做出来，但是就是卖不赚钱，这个现状多么打击团队的信心！

【手游路上跌进的坑】

1. 网游运营的巨大投入：如果只打算做一款大型赚钱的网游，这后续运营应该还是有精力投入的，但是如果只是为了试水，那运营肯定是要被拖垮的，而且网游只要不维护，流失率高的恐怖，基本就是直接死掉。
2. 人性游戏的诱惑：做一些擦边球的游戏，真的又轻松又有不错的回报，但是问题也是很直接，就是一旦被审查就肯定中枪，这东西能撑一时，肯定不能依赖。
3. 好玩不一定赚钱：移植了一个玩家口碑不错的游戏，最终效果就是玩家评论都挺好，但是游戏复杂度使得存留不高，最终结果就是下载量平平，付费率也平平。

4. 有用户基础不代表就一定有高下载量：也算是万万没想到，做了一个这么大众的游戏，居然输在下载量上，游戏还是看推广，或者是新创产品比较容易积累口碑。没有量真的什么都没有了。

5. 不要轻易冒险创新：创新比想象的难的多，哪怕是在成熟产品的基础上加一点创新，都不是容易的事，对游戏根本性元素创新几乎是在找死，做的时候总会遇到这样那样的问题。现在再看待创新，不是光鲜亮丽的与众不同，而是巨大的风险，有多少死在创新上的游戏，火起来的游戏也就是百分之一中创新成功的产品，初创团队不要拿前途去填那 99%的死亡比例。

6. 正视自己的能力：不管有多强的技术水平，宁可降低一个层次去做一个不费力的事情，也不要不讨好的去做力所不及的事情。刚出来想试试的心态总是在作祟，但现实会给你几巴掌让你正视的，还是一步步踏实的走，现在看到别人能不断扩充、做新产品，真心觉得佩服和羡慕，我现在只想让团队再多坚持做一个项目，但是现在也越来越感到困难，创业路上多走一步都是巨大的勇气和艰辛！

【到底怎么才是对？！】

对于一个还在爬坑的人来说，我真的不知道，从试错的经验和别人成功的经验来看，我觉得目前适合自己情况的项目，应该要具备以下几个条件。

1. 要有可玩性，不能指望同样的玩法就会有量：游戏本身好玩还是根本，这也是存留率的基本保障。但是这不是推动下载量的条件，下载量还是要靠推广来支撑，被淹没的好游戏应该还有很多很多。

2. 简单易上手：对于手游而言，越简单的越容易受欢迎，重度游戏一方面不是小团队的领域，另外一方面用户群数量也有限。把玩家想成操作上的小白，思想上的大师，让玩家稍微动一点脑筋感觉自己很牛逼，操作上还很容易达到目的。

3. 刺激消费:这个事情一直再被人们提及，要刺到消费者的痛处，但是具体怎么做又是很难拿捏的，初创团队应该还是选择成熟的办法，看别人成功的产品怎么做的，学习过来用。休闲游戏可以靠限制游戏次数来控制，打斗游戏靠玩家 pk 来刺激，人性游戏就靠人性弱点来吸引。

4. 先有一个赚钱的产品：创业要有长期的打算，一个项目走红真的概率太低，比较靠谱的就是做一个不用持续投入过多精力的产品，然后旧产品养团队，再开发新产品，这样比较容易进入良性发展。

5. 选对项目，别轻易创新：多花点时间在项目选型上，真的很关键，反复出错太受打击了，创业初期快速复制成熟项目可能反而比较好，先能保证有一定的现金流。

6. 搞定渠道保证下载量：能预先解决渠道是最好不过的了，再其次就是确定产品做出来会有渠道愿意推，最万能的办法就是做一个赚钱的产品再去找渠道合作。创业面对的是一个纯商业的环境，没有那么多所谓的朋友，都是利益上关系，能给别人更多利润就成了推动发展的根本。

7. 运气 :我真的不知道在策划时认为啥都具备了，但是出来还可能失败，这一切只有失败了才能知道错了，然后具体错在哪是不是能看准也不好说，我现在真的知道自己错哪了？！

一个优秀 IT 系统管理员该有的良好习惯

作者： 汪明 来源：http://mingwang.blog.51cto.com/1997299/1352180

做 IT 系统管理员有一阵子了，在各大博客和论坛都走访过，看过各式各样的技术帖子，深深地有感而发：

可能是因为第一份系统管理员的工作环境的熏染，如果我的观点有任何不合理的地方，我愿意听大家的意见。

1 我不用 Administrator！Domain Admins！

我看到了太太多的技术文档，部署教学，里面 10 篇文章 9 篇文章在用域的 Administrator 部署应用，我不太相信你们所工作的环境也是使用此账号来部署应用，因为这不合理也不科学，这个最高权限的账户通常是不需要使用，连密码可能也是放在了某个 RMS 加密后深层再深层的文件夹中。因为这个账户权限太高，风险太大。

那么如果你不是使用这个账户在真实环境部署系统，请不要发这种博文来误人子弟。

所有的应用全部应该是使用普通的专用域账号来部署，后面还会说道。如果真的是需要 Domain admins 这种权限 ,我也会在文档中明确说明 ,比如 Exchange 的部署需要标准的三大 admins ,Enterprise Admins，Domain Admins，Schema Admins。大部分的系统的部署只需要普通的 Domain User 就够了。

2 先关闭防火墙！

好吧，我现在的公司也是默认关闭了所有服务器的防火墙，但是我不推荐，非常不推荐。公司的策略不是我能更改的，我不知道前人是如何得出关闭防火墙这个结论。

很多公司默认服务器是需要开启防火墙的，我相信很多新人参照文档部署的时候发现这个服务器连不上，那个数据库无法访问，尤其是应用角色和数据库分开部署的时候。然后这个时候去咨询博主，博主回复到，先检查一下数据库服务器的防火墙关了没，那我的服务器不能关防火墙怎么办？

在我的文章中所有的防火墙都不会关闭，我会明确的告诉大家，需要开启防火墙的哪些端口。连接数据库需要哪些权限和角色。哪些应用，哪些服务通过那个端口做连接，这是系统管理员应该了解的，你可以多花几分钟，就可以在博客中告诉新人，省去了很多新人自己撞墙的麻烦。

3 自己的域账号做管理员！

好吧，我又不得不承认，我的新公司也是这样要求我的，我也不做任何评论。

所有的系统部署账号和管理账号都不应该是一个普通员工的域账号，而应该是一个特殊的固定账号或者用户组，比如 XXadmin,XXXADMINS 等，最最简单的原因就是当这个员工离职，不会对任何系统产生任何影响，只需要修改更新特殊管理账号的密码即可，而有新人加入企业后，也可以使得权限的添加变得更加容易。所有我自己在部署任何应用都是使用一个或者多个特殊账号。

4 Allinone 和勾选所有功能！

这个我看到的最多的地方就是在有 SQL 相关部署的地方，一上来就告诉新人，安装 SQL，勾选所有功能，然后把这个安装账号加到数据库的 sysadmin 中。比如我看到过的 SharePoint 的部署文档，写到，勾选安装 SQL 的所有功能，然后将用来装 SharePoint 的账号加到 sysadmin 中，这么做没有任何错误，我只是觉得会误导新人，实际应该是只勾选 SQL 的 DBengine，如果需要 SharePoint 有 Report 服务，也勾选 Report 服务，且如果是群集环境，那么 Report 是不支持群集的，需要单台工作。安装 Sharepoint 的账户需要是 SQL 数据库中 DBcreator 组和 Securityadmin 组中即可。我们既然写文档来教别人，就应该教的仔细一些，我之前得领导对我写文档的要求只有一个：假如我写 Exchange 的部署文档，那么就算看我的文档的人连 Exchange 是什么都不知道，也可以照着文档一步步将环境搭建完毕而不会出现问题。

至于 Allinone（All Feature in one machine）我认为怎么至少也要把数据库和应用分开吧？allinone 是为了项目测试某些功能而临时搭建一台才会这样，超过一个月的测试或者真实环境都不应该 allinone，而且多服务器环境与单台环境截然不同，只会 allinone 你是没法了解这个系统的全部功能的。

5 保持默认配置直接下一步即可！

我同意大部分系统部署过程中或者部署完毕后，很多配置使用默认配置是没有问题也是正确的，但我认为至少要说明和理解这个位置，这个勾是做什么用的，为什么勾着，不勾又会怎么样，在文档中说明，不然新人部署的时候也完全不理解。

一时半会儿就写了这么多，有空再补充吧

创新团队中常见的几种“怪人”

作者：翟胜军 来源：http://zhaisj.blog.51cto.com/219066/1352165

什么样的企业文化适合于创新人才生根发芽，什么样的团队机制适合于创新人才发挥潜力？创新不能只是领导喊的口号而已，要落地，要出东西，就需要打造优秀的创新团队。

创新团队之所以能存在，是因为找到了适于创新人才生长的土壤，但仅仅有土壤还不够，还需要有些特殊的组成“基因”，有了他们的带头，新意才会层出不穷，精品才能目不暇接。有了乔布斯的苹果公司才充满了创新的活力。看过乔布斯传的人，都会觉得他本人就很不寻常，执着、难以相处、独树一帜…但我们喜欢乔布斯，大众喜欢乔布斯，因为他为我们带来无数的惊奇。我们很难分清，苹果公司的品牌价值究竟是乔布斯呢，还是那个被咬了一口的苹果。

美国的社会机制、金融体系都比较适合于创新型公司成长，是新技术发芽的好土壤，其中很大原因，也许是因为他们能与这些“怪人”共存。Tom Kelley 在《创新艺术》一书中描述了 IDEO 公司的种种创新经历(无所不在的鼠标就诞生于这家公司)，也提到了创新团队中需要有八大类疯狂人，有很好的启示作用。

通过对国内企业内各种团队的观察，结合了中国人内敛的性格，我们认为，想要让团队具有创新意识，应该善待下面几类“怪人”：

1、追梦者：人只有在追逐自己梦的时候，常常执着到让人吃惊的程度。能够从喧闹的现实中抽身出来，不受外界的干扰与诱惑，一门心思地去实现自己的梦，创新需要这种不顾一切都劲头。许多追梦者还有很强的个人魅力，可以吸引很多人与他一起去追梦，但他们不是靠精彩的演讲，而是靠对目标的一致。说他们“怪”，是因为他们往往除了“梦”，很少关注其他的事情，没有了生活的柴米油盐，没有了朋友的闲聊娱乐，成天泡在工作室了，不修边幅，没有假日，好像是一些远离人群的人，甚至充满了神秘感；

2、异端者：创新往往源自怀疑。打破常规才能发现新思路，破除旧规则，才能发现新天地。每个组织都需要某种对立面，即挑战现状者。这些人往往是一些“少数派”，是难以合群的异端，他们对惯例与规矩、领袖与权威都不买账，常常另辟蹊径，以绕过规则、追求自由为乐。说他们“怪”，是因为他们常常是普通团队中的“捣乱分子”、“不受欢迎的人”，或者是经常给领导带来麻烦的那类人；

3、吸尘器：创新团队为了不束缚大家的思想，管理都比较松散，没有统一的制度、严格的作息时间。但混乱不等于无序，能够在会议中纠正跑题的讨论，让工作方向迅速回到正确的轨道上来，能够处理各种棘手的人际关系，协调好紧张的资源分配，能够合理搭配各种性格、秉性的人员，成为高效的课题小组…这些需要清晰的思路和敏捷的反应能力，以及干练的做事风格和超强的执行力，这类人在企业中，往往被用作处理棘手问题的“空降兵”，替领导扫清麻烦的“特种兵”，所以也称作“企业吸尘器”。吸尘器最大特点就是职业化、条理化，办事情从不拖泥带水。说他们“怪”，是因为他们“缺乏”人情味，有时冷酷得甚至很令人讨厌；

4、工匠：再好的创意要变成现实也需要有动手的人。能够充分领会设计者的意图，巧妙地实现设计者的精心构思，并最终呈现给用户一个精美的作品，工匠往往是团队中勤勤恳恳、不善言辞的那些幕后人。说他

们“怪”，只是因为他们常常被人们忽视，他们的能力究竟有多大，没人清楚，尤其是那些层出不穷的小窍门，数不胜数的小工具，好像是每每给 007 提供各种新式武器的那个仅出现几个镜头的怪博士。他们都像是有双神奇的手，把别人脑海中的虚拟影像，迅速变成现实中的精美艺术品；

5、瞎打听：这类人常常不好好“工作”，无时无刻不在找人聊天，经常是忙着参加各种聚会与沙龙。他们不仅熟悉业界的最新动态，知晓同行与对手的最新进展，而且知道无奇不有的花边新闻，当然也常常沉迷于各种奇思妙想的收藏…总之，你从他这里可以知道你想要的任何信息，你想设计一个最佳实现的用户方案，应该先问问他知道不。说他们“怪”，是因为他们精力实在旺盛到令人吃惊的程度，异常热情，容易兴奋，总让人有一种难以推脱的感觉。没人能明白，他们怎么会有多少朋友，那么复杂的人际关系，他们是怎么搞清楚的；

6、万金油：这种人能力好像不是很突出，但常常是充满激情，自学成才的多面手。他们熟悉各种技术，了解非常多的行业知识，有取之不尽的各种经验。很奇怪的是，他们常常在大公司中找不到合适的职位，干什么都不长久，学什么都不深入，牢骚怪话不断，常常被认为是“一条鱼腥了一锅汤”的那条鱼。说他们“怪”，是因为你发现正统方法走不通时，他们常常知道一些可行的“邪门歪道”。对什么都“不求甚解”的“陋习”，反而让他们敢于想象，乱中取胜，善出奇招。真的是不求甚解吗？好像他们了解很多我们所不知晓的、不常见到的“奇怪”知识。

这六类“怪人”，常常是普通团队所“不喜欢”的人。在大多数的领导眼里，他们不是人才；在大多数的员工眼里，他们是怪人，是难以相处成朋友的人。

作为一个创新团队，发现并收罗这六类“怪人”，是很有必要性的。但值得注意到是：要提供一个环境，即让他们互不干扰，又能紧密合作。最为重要的是，与他们能够长期愉快合作。你应该设法引导追梦者，让他们的梦与团队的目标合二为一，并给予他们必要的基础条件；你应该包容那些异端者，让他们充分地表达，并给予探索、证明他们自己的机会；你应该谨慎使用吸尘器，消防队员应该适时地出现在救火现场，而不是休闲的假日海滨浴场；你应该善待那些工匠，给予他们充分的尊重，尤其是那些匪夷所思的嗜好与收藏；你应该允许瞎打听有一定的自由度，严谨的工作时间常常让他们如坐针毡；你应该容忍万金油的牢骚与怪话，偶尔的小恶作剧也可以活跃一下团队的气氛，还记得鸡鸣狗盗的典故吗？

每个人都有自己的优势，“怪人”不同于常人的思维，往往是创新意识的灵感来源。当然，人常常是各种性格特点的复合体，很少有单一性格极端的人。善于用你的眼睛发现他们，是与他们合作的前提。

写给同事的一封信

作者：李云 来源：http://yunli.blog.51cto.com/831344/1348721

亲爱的同事，

转眼我在这个团队工作已有一年的时光，这一年也完成了我从通讯行业转入互联网圈的过渡。过去的一年给了我很多观察（团队）的机会，也带给了我不少思考，从我过去一年的寥寥几篇博文你应当能看到部分。

今天，我想借这篇文章与大家聊一些内容，以便你更加明白：为什么我在工作中对自己和大家的要求都那么高？为什么我强调责任与重视培养工作好习惯？为什么我会直接批评和积极表扬人与事？希望你的其它“为什么”也能在这里找到答案的线索！

现实与虚拟

现实社会中有很多让人恼火的事：想排队按序办事，却总有些人插队；想维护家里楼梯走道的卫生，却发现总有人乱扔垃圾和随地吐痰；车祸明明缘于对方过失，但他却百般否认与狡辩；想喝上干净的自来水，不求助于净水器似乎没有可能；等等。林林总总！

对于一名构建虚拟世界的软件工程师，我们不得不变得“性格分裂”，因为我们不能将现实中的脏、乱、差带到我们所构建的虚拟世界里。这种“不能”并非我们一厢情愿，而是现实所迫，因为“脏乱差”的虚拟软件世界一定会给用户带来糟糕的产品使用体验，也会为我们的工作生活增加额外的成本（加班等）。一旦明白这一点，你就会理解我为什么在工作中的要求会那么高，而且是多方位的！

面对现实与虚拟，我们不得不适时进行模式切换。在社会生活中，不要过于较真而影响自己的健康；在工作生活中，我们应力求构建完美的虚拟软件世界。

刚加入团队之时，发现大家所写代码多了些随意，团队知识管理也没有“官方”途径而是各自写在自己的文档或记在头脑中（那时作为新人的我还是为此经历了不必的痛苦）。为了改变编码随意这种现状，几个月前我决定着手实施编码规范。坦白说，这是我的职业生涯中首次大张旗鼓地推行编码规范，以前我一直认为写出有质量的代码（和文档）是软件工程师所应做的本份之事，无需过于强调。当时决定推广的另一大主因是，我们产品所基于的开源项目的根基非常好，除了自身代码就是一个优秀的参照外，更有着完备的编码规范和规范检查工具。然而，编码规范的最高境界并非“格式”，而是我们的“态度”，因为规范解决的只能是形式问题，而无法规避不恰当命名等非形式问题。也正因如此，你们所写的大部分代码我都会走查，通过不断指出问题并帮助改善的方式培养大家的认真态度。最近，每当我看到代码中存在你们改善质量的痕迹时，都会会心一笑；面对大家指出我所写代码中所存在的改善点时，我更加高兴并给予肯定和感谢，因为我坚信这是我们团队所应倡导的文化。大家回头看一看，过去的几个月我们团队在这方面取得了质的进步。

谈及团队知识管理，不得不说到我所带头编写的《软件开发指南》和《技术白皮书》。《软件开发指南》中的内容一方面很好地指导了我们的工程实践（涵盖软件设计、流程等大大小小的各方面），另一方面为新人上手起到了积极作用。前者从目前我们的项目中基本杜绝了模块混乱、加速了新版本合并速度和使得软件开发活动更规范化可以看出，后者则从最近 WL 在晨会上说“《指南》写得很好”得以佐证。

值得强调的是，所有这些进步都是我们共同创造的，没有你们的积极参与和快速跟进根本不能取得现有成绩，也很难轻松走得更远。还记得我在项目总结会上所说的“我为人人，人人为我”吗？

现实如此不完美，我们能否从构建的虚拟软件世界多找到些完美呢？！

面子与责任

面子的重要性无需多言，“捍卫面子”的意识也深入了我们的骨髓。然而，正如我曾在邮件中所提到的，以我在外企工作时的观察发现，美国的工程师之所以更富质效在于他们不是将面子放在首位，取而代之的却是责任。也正因如此，美国的工程师很喜欢发问且有时的问题让人觉得 “尖锐”，这一特质无论他们是面对同行、还是“领导”都一致。

我坚信，一个讲面子与一个讲责任的团队将形成截然不同的两极。讲面子的团队大多低效并很有可能集体无能，而讲责任的团队将运作得务实、高效；重视面子的团队看到问题采用的是回避态度，讲责任的团队看到问题会指出并较真甚至承担。责任之所以重要，是因为它会促使我们在工作中有所作为——用心按时、按质完成工作。一个讲责任的团队也不容易出现“技术军阀”和“管理官僚”，这两者对于团队的杀伤力都可称为是“核武级”。

重面子的团队还有另一个突出表现：团队成员很“听话”，上级说什么就做什么，不想不问，但承担不良后果。这种团队往往会凸显出“领导”的“重要性”，因为没有“领导”团队就不知要干什么、要向哪走。与之相反的是，重责任的团队即便短时没有领导者也能有序运作，甚至倒逼管理者有所作为。

另外，责任不应只是对于自己和团队，还有对于家庭的部分。比如，通过尽可能少加班多与家人在一起、陪伴孩子成长。意识到这种责任，你往往会花更多的时间去提高自己的能力和效率，而不会一味地想着加班是解决工作问题的万能药。我一直不能理解那些没事却在加班的人，他们对于家人如此，在工作中真的可靠吗？频繁的加班不是个人能力有问题，就是管理出现了问题。

再者，讲责任会促使团队的成熟，使得成员重视承诺，这对项目的有序运作至关重要。重视承诺的含义是：我们对于无法按期完成的工作不承诺，而一旦承诺就要努力达成。

理解了我对面子与责任的看法后，相信你能明白为什么我会在晨会上不时提问，也能理解为什么我敢说且主动承担非职务之内的（管理）工作、会私下找你聊工作上的事和很少加班。这一切都是责任使然，是我应该去做的！

批评与表扬

人追求完美是无极限的，无论我们多么有经验、专业和职位多高，始终能做一个更好的自己而获得成长。显然，成长的过程中不可避免地会犯错。矛盾地，在纠错的过程中我们又有惰性而阻碍前行。面对自身错误惰于纠正的情况下，我们如何向前？需要来自他人的批评

说到批评很容易让人想到《人性的弱点》中所倡导的“不要批评他人”观点，也很容易让人浮想“说话的艺术”。我对于批评有着不一样的看法和使用方法！

首先，我们得对批评的反应调低一点敏感度，不要一听到批评就什么都听不进，甚至出现逻辑混乱地狡辩的状况（比如，人家批评你的是事，但你说人家批评你时的态度不对。其实，只要人家事批评得对，即使态度有点不妥也得包容，可以将之理解为“我做错了而导致别人的情绪”）。其次，碰到批评先深呼吸，之后想一想所批评的问题确实存在吗？在一个讲责任的团队中，批评不光很少出现，一旦出现大家也能平常心面对（注：批评出现多了很可能表明团队管理出现了问题，不作为的事多了。从团队对于批评的态度可以看出这个团队是重面子的还是讲责任的）。在使用批评的方法上，我主张批评的目的不是让人难堪，而是帮助他人改进，在实施批评之前先友好地提醒对方错在什么地方和如何改进。如果友好提醒还解决不了问题，那只能实施批评了。被批评虽让人难过，但也会让人记忆深刻而避免下次再犯同样的错。对于批评

的艺术问题我并不想多谈，因为工程师大多很单纯、是非少，只要各自心态摆好真的无需复杂到将批评“艺术化”。

谈及批评不得不说说表扬。表扬是一种对他人付出的肯定与欣赏，但中国的工程师好象不大擅长使用这种方法去表达自己的情感。我发现表扬很容易出现“礼尚往来”的现象，你今天表扬了别人，过几天可能也会收到对方的表扬回报。这种事情如果在团队出现得多了就很容易带来轻松的氛围，也容易让人体会到自己对于团队的重要性，这样不好吗？

与表扬相似的是，我们在工作中还可以：当因自己的工作失误而给人带来麻烦时说声“对不起”；获得别人的帮助后道声“谢谢”；向他人咨询问题时用“请教个问题”开头。这些小细节做起来很容易，但对团队建设的贡献却很大！

出色的团队离不开批评，且是通过表扬而培养的！

质效与习惯

我在美国工作的（累计）半年时间里，所涉项目周末从未见人加班，即使项目非常紧张依然如此。周末行驶在高速公路上，时不时能看到房车或被拉着的游艇从车旁驶过，很是感叹。人家不加班又何以如此高质效且过着丰富的周末生活呢？

在软件行业 ,质量与效率是一个永恒的话题 ,但却鲜有人真正了解它们从何而来。也往往迷失于 SCRUM、CI、ET 等诸多方法论中。

要做到有质效地工作，首先离不开各位承担应有的责任，其次是良好的工作习惯。前者会促使我们持续变得更专业、善于思考和较真，后者则使我们高效，两者一结合质量也随之有了。以我的观点，中国的工程师只要没有将责任和工作好习惯落实好，谈其他的方法论都没用，谈了也白谈。

说到工作好习惯很容易让人觉得发虚，有点看不见摸不着的感觉。我也一时很难在这说清楚，需要大家在工作中多观察，了解我是如何做和思考的。培养工作好习惯的另一个值得一提的好处是，它有助于杜绝技术问题演变成管理问题，从而使得团队更加轻量和高效。

精品产品是有气质的，是团队责任与工作好习惯的折射！

最后，过去的一年我们一同收获了不少，在这个伟大的开源项目上工作也让我觉得很有乐趣。谢谢你在碰到问题时主动与我讨论、谢谢你曾经授予我的技术决定权，因为尽管你们有的不说，但我能感受到你对我技术能力的肯定和信任！同时，也谢谢你们曾经给予我的帮助！

从“网上说的能信么”说开去---学习的思考

作者：孙杰 来源：http://xjsunjie.blog.51cto.com/999372/1348446

网上说的能信么？”这句话对大家来说并不陌生，很多时候很多场合当你听到这句话的时候，却不得不思考一下。网上的说的东西该信不该信，思考之后相信我们自有决断。

一、经验来自于实践，但实践又是生动的、不断变化着的。

一次在客户现场，解决一个 DB2 的报错，报错信息如下：SQL0964C 数据库的事务日志已满。

DB2 数据库报事务日志满一般是指当前事务无法写入到活动日志中，主要是由于 DB2 主日志文件和辅助日志文件已经全部用完或者没有足够当前事务写入的空间所造成的。解决这个问题需要检查日志大小和主日志文件和辅助文件个数，并进行修改这三个参数来增大日志容量。分析完问题后，给甲方的技术经理发邮件，说明了一下问题原因及该怎么处理，没有具体解释原理，只是附了一个关于此问题描述的网页地址。甲方的技术经理很快回复了，说是网上的东西能信么，赶紧删归档日志，别让数据库宕了。我看了邮件，简直无语了，这都哪跟哪啊。听人说，这个经理以前是搞 ORACLE 的，怪不得会这样。在 ORACLE 中，当 ORACLE 归档日志满了后，将无法正常登入 ORACLE，还有可能会引起数据库挂起或系统宕机，需要删除一部分归档日志才能正常。他误以为 DB2 的事务日志跟 ORACLE 的归档日志是一回事，所以会这样反应。既然这样，还是找个权威的官方解释吧，然后再次发了一封邮件，附上了 IBM 官方网站关于这个报错的描述和解释。在 DB2 中，数据库事务日志已满不是由于磁盘空间满引起的，而是由于没有落实的事务总体过大，超过了数据库事务日志所能容纳的最大大小所造成的。这回甲方的技术经理终于信了，一切按邮件里所说的步骤去处理，问题很快解决了。

这件事告诉我们，经验不是万能的，要结合具体的实践。当具体实践发生了改变，原有的经验可能就过时或失去了指导意义。另外，工作中如果遇到这样的问题，尽量使用官方的解释来佐证自己的思考和判断，这样更有说服力。

二、拿来主义不能解决一切问题，尽信书则不如无书。运维中时常会面临各种各样的挑战和难题，很多时候在你的能力范围内，什么招都试了还是没有办法搞定问题。这个时候，我们通常会百度、GOOGLE 一下，看看网上有没有类似的问题，如果有或者恰好跟你的问题一模一样，那很幸运可以直接拿来用一下，快速解决问题。然而多数情况，有类似的问题然而解决的方法却各异，那你就要结合自己的具体工作环境和问题发生的原因，去辩证的思考了，所以拿来主义并不能解决一切问题。

举个例子，在 ORACLE 中有个错误 ORA-06553，在官方文档中也查不到，那你又该信什么。引起这个错误的原因却有很多种，像这样的问题只有结合具体的情景，靠自己去分析思考和解决了。

公司曾有一套 ORACLE DG 数据库,其中有一台 primary 主机因主板硬件有问题突然坏掉了，发现后将备库 standby 切成了主。坏掉的主机在换了主板后，重装了 ORACLE，然后从备库把主库恢复回来，具体恢复过程在这里就不细说了。两个库起来后日志传输正常，使用 sys 用户登主库，也没发现什么问题。过了一阵子，公司开发部门的工程师告诉我客户端用户连不上数据库。我登上主机试了一下，发现只有用 sysdba 登的时候没问题，查询数据也能查出来，而其他用户连都报错。报错信息如下：

```
ERROR:
ORA-06553: PLS-801: internal error [56319]
Error accessing package DBMS_APPLICATION_INFO
ERROR:
```

ORA-06553: PLS-801: internal error [56319]

查 MATALINK,官方也没什么说明，去网上搜一下，有人说是因为用户 ID 不同造成的，于是核对了一下两边 oracle 用户的 id，发现 DBA 组的 ID 不同，修改了一下再连仍然报错！

这个时候该怎么办，否定排除了这个原因，我们还要继续思考，再搜再试。

接着又发现有一篇帖子说从 32 位库移植到 64 位容易出现这个问题，那就执行一次从 32 位到 64 位数据库的升级操作吧！反正还有备库，实在不行切了。使用 Startup upgrade 升级后，再进行 shutdown immediate,再启再执行 utlrp.sql 脚本把相关内容在 64 位平台下编译一遍，最后再试试，发现其他用户也很顺利的进去了，再无此类报错了。

这个问题告诉我们，即使遇到了官方也无解释和说明的问题，除了用百度、GOOGLE 去搜索，我们只有自己不断地去尝试去分析去思考，相信自己才能力克难题。

三、学习与实践，取其精华去其糟粕，一句话贵在坚持思考与总结。

在工作中，其实每个人都会遇到这样那样的难题和挑战，我们只有凭着一颗不断探索和求知的心去学习、去思考和实践，才能解决问题。同时，在不断学习的过程中，取其精华去其糟粕，形成自己的一套知识体系和思想，并不断完善和提升自己。引用大卫张的一段话“虽然没人真正愿意一年经验用十年，但是一年经验用十年却是一个让人很悲哀的客观现实。其根本原因和最大证据就是，大多数人已经不能从工作中学到新东西了，已经不能获得新经验了。”究其原因，我们失去了较真的态度和探索的精神，我们没有在实践中持续的思考与总结，我们习惯了那样悠闲安定的生活，这样渐渐地不自觉地让自己掉队了，直到最后我们不得不重新寻找自己的奶酪。

也许这样的情况正发生在你的身上，想想你已经多长时间没有学习新东西了，工作无非就是走流程、写代码、调试问题，日子一天又一天不断的重复，想想都让人感觉可怕，难道一辈子就这样了么。不少人曾努力过，想改变现状。或者是换公司或者接受外部培训，看看外面的世界是否不同；或者开始削尖脑袋，一定要加入到公司重要的项目去。但是你忘了，只有改变自己才能改变世界，当你静下心来审视自己，更加深入的分析和思考问题，你才能获得更深的感悟和与众不同的认识。

陈云说“不唯上，不唯书，只唯实。" 就是告诉我们不迷信权威，不盲目照搬书本，只有从实际出发，实事求是地研究处理问题，才是最靠得住的。

至于网上说的能信不能信，只有我们擦亮眼睛，带着一颗思辨的心，去分析去研究，相信思考之后我们自有决断。

编后语

《51CTO 博客月刊》为 51CTO 博客整理出品

如果您有意投稿，请联系我们

如果您有反馈意见，请告诉我们

联系方式：

Email：blog@51cto.com

QQ: 1173854158

欢迎关注我们：

@51CTO 技术博客